8x8 Application Note

SYMPHONY VoIP ACCESS GATEWAY

8x8 Confidential

Document No. 000-0040-001, REV. B

8x8 INC.

Copyright Notice

Copyright ©1999 8x8, Inc., 2445 Mission College Boulevard, Santa Clara, California 95054, USA.

-- ALL RIGHTS RESERVED --

This document is copyrighted © 1999 by 8x8, Inc. No part of this specification may be reproduced transcribed, transmitted or stored in a retrieval system in any part, or by any means without the expressed written permission of 8x8, Inc.

Trademark Acknowledgment

8x8, Audacity, Kodiac, VCPex, VCP, LVP and VPIC are registered trademarks of 8x8, Inc. G.711, G.722, G.723, G.726, G.728, H.323, SGCP & MGCP refer to the International Standards. All other trademarks are trademarks of their respective companies and are used for identification purposes only.

Disclaimer

8x8, Inc. makes no representations or warranties regarding the content of this technical specification. This information is preliminary and is subject to change at any time without notice. 8x8, Inc. assumes no responsibility for any errors contained herein.

Patents

The product(s) included with this guide may be covered by one or more of the following patents:

US 5,339,076
US 5,351,208
US 5,379,351
US 5,594,813
US 5,761,280
US 5,790,712
US 5,793,984
US 5,898,898
US 5,901,248

Additional patents are pending.

The customer should notice that in the field of audio compression and VoIP telephony there are a number of patents held by various parties. It is the responsibility of the user to assure that a particular implementation does not infringe on those patents.

Contents

Confidential

Section 1 - Introduction

The Symphony VoIP Gateway is a low complexity design which bridges the gap between conventional telephony systems equipment such as telephone and fax machines and the new world of Internet protocol (IP) communications over private networks and the public Internet. One Symphony board can connect up to four independent calls simultaneously.

The Symphony board features the 8x8 Audacity Internet Telephony Processor, which provides the DSP and command/control processing required to compress the audio and format the call for the transmission over IP networks. Connection to IP networks is made via an Ethernet MAC/PHY chip, which provides access to 10BaseT Ethernet and manages flow control. Specified with low material cost and high audio quality in mind, the Symphony VoIP Gateway is a complete solution for connecting existing telephony equipment in homes and small offices to broadband networks.

The design may use either the SGCP/MGCP or H.323 standards for VoIP, and it is fully compatible with the Cablelabs "Packet-Cable" initiative, with the emerging H.GCP standard and with Microsoft NetMeeting. The Symphony board incorporates FLASH memory and remote upgrade capability, so systems may be programmed with updated protocols via the network.

The Symphony Gateway is available in four forms: a populated and tested PCB, complete with Codec software for integration directly into product, as a unit level assembly which includes the PCB in a housing, as an evaluation system with hardware and software tools, and as a developer's kit. The developer's kit includes schematic files, layout files, and the software libraries. The audio Codec libraries for the Symphony board are supplied as object code. A comprehensive command/control and GUI application is supplied as source code with the developer's kit, enabling the rapid modification and customization of the design for the ultimate in development flexibility.

Standards

The Symphony board supports the H.323, MGCP & SGCP communication standards for IP networks.

- H.323
- SCGP, MGCP
- DOCSIS 1.1 compliant
- MS-NetMeeting compliant
- Supports DHCP, TCP/IP, UDP, SNMP

Audio performance

- G.711 A-law
- G.711 μ-law
- G.723
- G.726
- G.728
- Acoustic echo cancellation (64mS tail, 18dB rejection)
- Voice Activity Detection
- Comfort noise generation
- DTMF detection
- Call Tone generation

Network Interface

- 10BaseT

Host interface

The Audacity chip on the Symphony board utilizes the Audacity host port to interface to the developer's connector. Developers may connect a PC to the unit via a board and cable provided by 8x8 for debug and software download.

Product kits

The following Symphony Gateway products can be directly ordered from 8x8.

- Board level assembly
- Unit level assembly
- Evaluation kit
- Developer kit

The board level assembly is supplied as follows.

- Mainboard
- Power supply

The unit level assembly is supplied as follows.

- Mainboard in enclosure
- Power supply
- Installation manual

The evaluation kit is supplied as follows.

- Mainboard in enclosure
- Power supply
- Installation manual
- Application document
- Quickstart card
- Software configuration tools
- Hardware configuration tools (STIC card)

The developers kit is a complete design package for the Symphony board. It provides all of the information needed by hardware designers, software engineers, product test engineers and component engineers to develop and manufacture a residential or SOHO IP Telephony Gateway for connection to ADSL or Cable modems. The contents of the kit are:

- Two Symphony Mainboards in case
- Power supply
- Installation manual
- Application document
- Quickstart card
- Software configuration tools
- Hardware configuration tools (STIC card)
- Schematic diagrams
- Gerber and Job files
- Full component specifications
- Bill of materials
- Programmable component files
- FLASH ROM images
- Diagnostic software source code
- H.323 and MGCP object code libraries
- VoIP Gateway application source code
- Audacity "C" compiler and tool kit
- API documentation

Confidential

Section 2 – Application

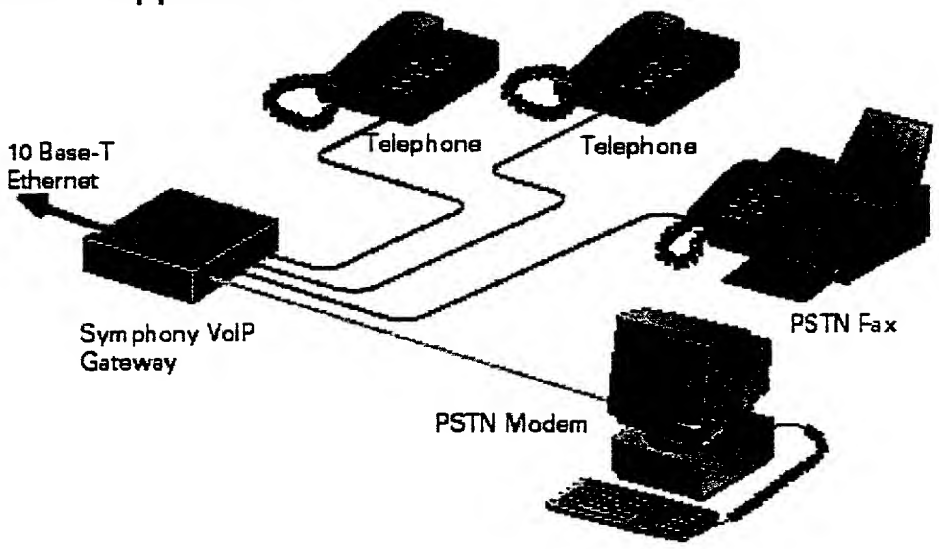


Figure 1: Symphony Gateway architecture

Product Application

Figure 1 shows a Symphony VoIP gateway application. Attached to Symphony are two telephones, a PSTN Fax machine and PSTN Modem.

Symphony behaves as a four line telephone switch with resident Ethernet gateway. Symphony supports all features expected from a commercial PSTN switch provider such as:

- BORSCHT (**B**attery, **O**ver-voltage, **R**inging, **S**upervision, **C**odec, **H**ybrid and **T**esting)
- Caller-ID
- Three-way calling
- Detect DTMF
- Call Waiting
- Last number redial
- Call return

It is also possible to ring one phone from the other on Symphony. A major advantage of the Symphony is the built-in Ethernet gateway, which facilitates a connection with other gateways via an Internet connection.

Functionality

Symphony provides full PSTN compatibility for four lines. Inside each Symphony gateway is a powerful 8x8 Audacity ITP, which codes a high-quality digitized audio steam into a low bit-rate data stream. Audacity then packetises the coded audio and provides IP stack support. The resulting data packets can be transmit easily over an ethernet connection to the Internet.

Supported Equipment

Symphony is capable of supporting four PSTN devices simultaneously. These 4 PSTN devices can comprise any combination of telephone, fax machine, modem or other common PSTN device.

Connections

The PSTN apparatus is connected to Symphony using US telephony wiring via RJ11 connectors. Non-US Customer Premise Equipment (CPE) can be attached, using readily available adapters. Symphony is connected to a local Ethernet connection via a standard RJ45 connector. A low voltage, dual conductor cable supplies power via a 1.3mm jack socket.

The 10BaseT interface

Symphony supports the IEEE 802.3 10BaseT interface.

Telephone ringing

Symphony supports up to 4 CPE. Symphony is capable of ringing all phones simultaneously. The ringing voltage can only be supplied to one line at any instant; Normal ringing "cadences" allow the ringing of any line to be concurrent with silence on the other lines. This is shown in the diagram below. Ringing cadence can be changed to allow for Distinctive Ring and standard national ring patterns.

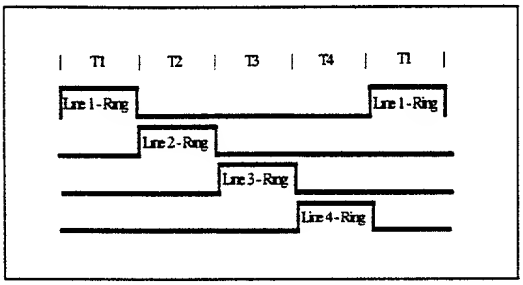


Figure 2: Ringing Pattern

Status indicators

There are a number of status indicators to show the various states of the Ethernet connection and the four PSTN lines. These are described in more detail later.

System Control

Symphony can be controlled by a control data sequence received through the Ethernet port or by DTMF detection from a CPE device connected to one of the PSTN lines.

Communications Profiles

The Symphony Gateway may use either MCGP or H.323 communication standards, depending on profile and end use of product.

MGCP

This recommendation describes the Media Gateway Control Protocol (MGCP) for use in a centralized call control architecture and assumes relatively simple client devices. This protocol is used for controlling voice-over-IP (VoIP) gateways from external call control elements. MGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and is handled by external call control elements.

H.323

This recommendation describes terminals and other entities that provide multimedia communications services over Packet Based Networks (PBN) which may not provide a guaranteed Quality of Service. H.323 entities may provide real-time audio, video and/or data communications. The PBN over which H.323 entities communicate may be a point-to-point connection, a single network segment, or an inter-network having multiple segments with complex topologies. H.323 entities may be used in point-to-point, multi-point, or broadcast configurations.

Audio Processing

The application program includes an audio codec, DTMF tone detection and Acoustic Echo Cancellation (AEC). These are among the basic building blocks needed to make a VoIP product.

Audio CODECs

The following CODECs are supported:

- G.711 A-LAW PCM 64kbs, 8kHz sampling - 8 channels
- G.711 μ-LAW PCM 64kbs, 8kHz sampling - 8 channels
- G.723 MPCMLP 6.3kbs, 8kHz sampling - 4 channels
- G.726 ADPCM 16, 24, 32, 40kbs 8kHz - 4 channels
- G.728 LD-CELP 16kbs, 8kHz sample rate - 2 channels

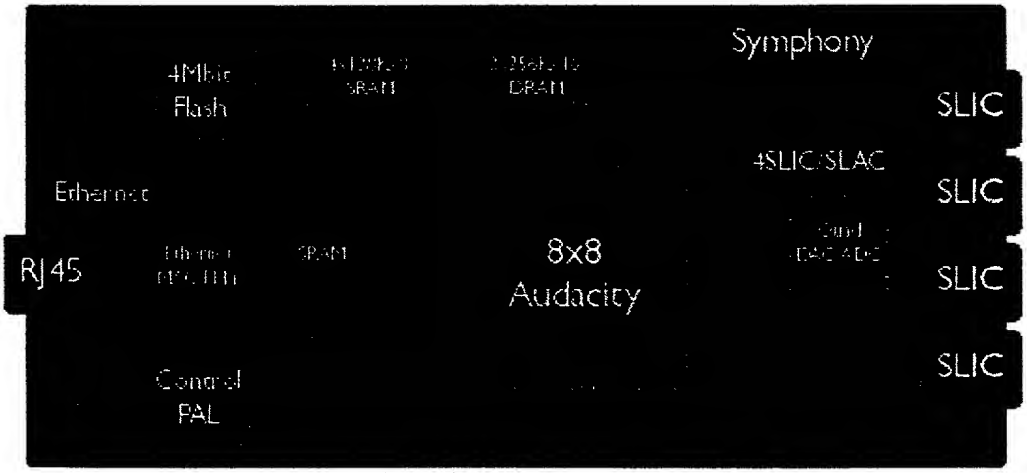


Figure 3: Symphony Block Diagram

Hardware Overview

Figure 3 above shows the Symphony architectural block diagram. The main features of the design are the Audacity Internet Telephony processor, memory sub-system, audio sub-system, I/O sub-system, quad SLIC interface and related circuitry.

PCB

The Symphony Gateway is a single board design, with the PCB measuring 5.9 by 7.1 inches. Four mounting holes are located on the board, one in each corner. The PCB is a four layer design.

Connectors & Indicators

All input / output connectors are located at the back of the PCB

- Four RJ11 connectors
- One RJ45 connector
- Power supply connector

All status and user interaction features are located at the front of the PCB.

- Line 1 through 4 status LEDs
- Call Agent ready LED
- MGCP Link Status LED
- System Status LED

System control

System control is achieved in the Symphony Gateway by the resident 8x8 Audacity ITP processor.

Memory

The mainboard has three types of memory used for program storage, execution and caching of data.

- FLASH Memory
- SRAM Memory
- DRAM Memory

The FLASH memory is non-volatile and is used for program storage and user settings when the power is off. The Flash is configured as 512k x 8 bits. For flexible erase and program capability, the 512kbits of data is divided into 11 sectors: one 16kbyte, two 8kbyte, one 32 kbyte, and seven 64 kbytes.

The SRAM memory is volatile and is used for program storage and execution when in operation. The SRAM is configured as one bank of four IC's. Each IC has eight data bits and 128k of total storage. When the IC's are mapped together they provide 128k x 32 bits (512k x 8) of contiguous storage area. Each SRAM IC has an asynchronous access speed of 12nS or better.

The DRAM memory is also volatile and used for general data storage and caching of incoming Ethernet data. The DRAM is configured as one bank of two IC's. Each IC is sixteen bits wide with 256k words of total storage. When the IC's are mapped together they provide 256k x 32 bits (1M x 8) of contiguous storage area. Each DRAM IC has an access speed of 50nS or better. Both FPM and EDO DRAM are supported.

SRAM Bus
The SRAM bus used in the Symphony Gateway is a 32 bit parallel bus and runs between the Audacity processor, the SRAM, FLASH, ethernet controller and the EPLD. The bus has separate WRITE, READ and four byte ENABLE signals, all active low.

DRAM Bus
The DRAM bus used in the Symphony Gateway is a 32 bit parallel bus and runs between the Audacity processor and the DRAM. The bus has separate WRITE, READ, RAS, CAS signals, all active low.

ETHERNET MAC / PHY
The ethernet controller used in the design is a Fujitsu MB86964. The device supports the 10BaseT & 10Base2 interfaces. This device is mapped onto the SRAM bus.

CONTROL PAL
The Symphony glue-logic is based on a Lattice Semiconductor core. Lattice can supply a suite of related tools allowing both design and in system programing of this part. The specific part used is an ISPLSI2128V-80LT100. This is a 3.3V I/O part.

The Lattice chip is programmed with glue-logic designed to suit the existing profile of the board. However, if the board has a different function or application to meet specific needs, new glue-logic may be needed to support the external hardware features from software. For a full description of the lattice design, please refer to tables 1,2,3 & Appendix 2-Glue logic.

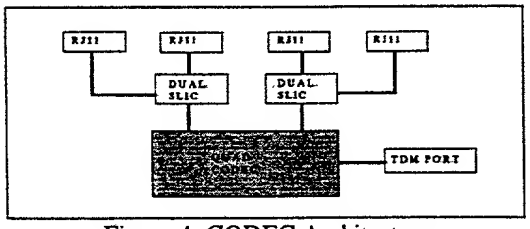


Figure 4: CODEC Architecture

PCM CODEC SLICs

Two dual SLICs provide the four PSTN ports. Each port is identical to a single PSTN subscriber line and can be used in the same way. PSTN devices such as telephones, facsimile machines and modems can be connected.

Architecture
The architecture of the PSTN interface is based upon two dual SLICs feeding a single Quad CODEC. Four separate channels can be digitized in this way. This is shown below.

Quad CODEC & Dual SLIC
The quad codec used in the design is a Lucent T5504 / 7504 (U11). It is capable of coding and decoding 4 channels simultaneously. There are two dual SLICs (U7 & U10) which extract the analog audio from the PSTN lines and pass to the Quad CODEC. The part used for this is the Lucent L8576.

PSTN Interface

Attached to the PSTN connector are a series of components designed to protect both the telephony equipment and the SLIC from undesired operating conditions such as voltage spikes. Much of this conditioning is performed inside the Lucent L7591 SLIC chip.

The SLICS have resistive impedance, and are best suited for driving short-haul local loops, commonly less than 5000ft. Each line is capable of supporting the 5000ft individually. The SLICs are designed with a minimum power dissipation occurring at a local loop length of 500-1000ft.

Battery Supply
To power the CPE attached to the PSTN lines, a battery supply voltage must be present, since CPE devices rely on the local central office to supply power. As Symphony performs the role of a small central office, it must provide a standard PSTN power supply to each of the connected devices. Each connector has a REN rating of 5 for short-haul applications, primarily within a home or small office.

TDM Interface
The Time Division Multiplexed (TDM) port allows access to the high-speed, bi-directional serial bus which is used to transfer the PCM encoded bit stream between the PCM codec and Audacity ITP. The TDM interface on the Audacity ITP can implement a number of high-speed serial protocols including CHI, GCI, K2, SLD, MVIP and IOM2 formats. The TDM port can also act as a general purpose 16Mbit/sec serial link.

Confidential - Lantronix

Chapter 3 - Hardware

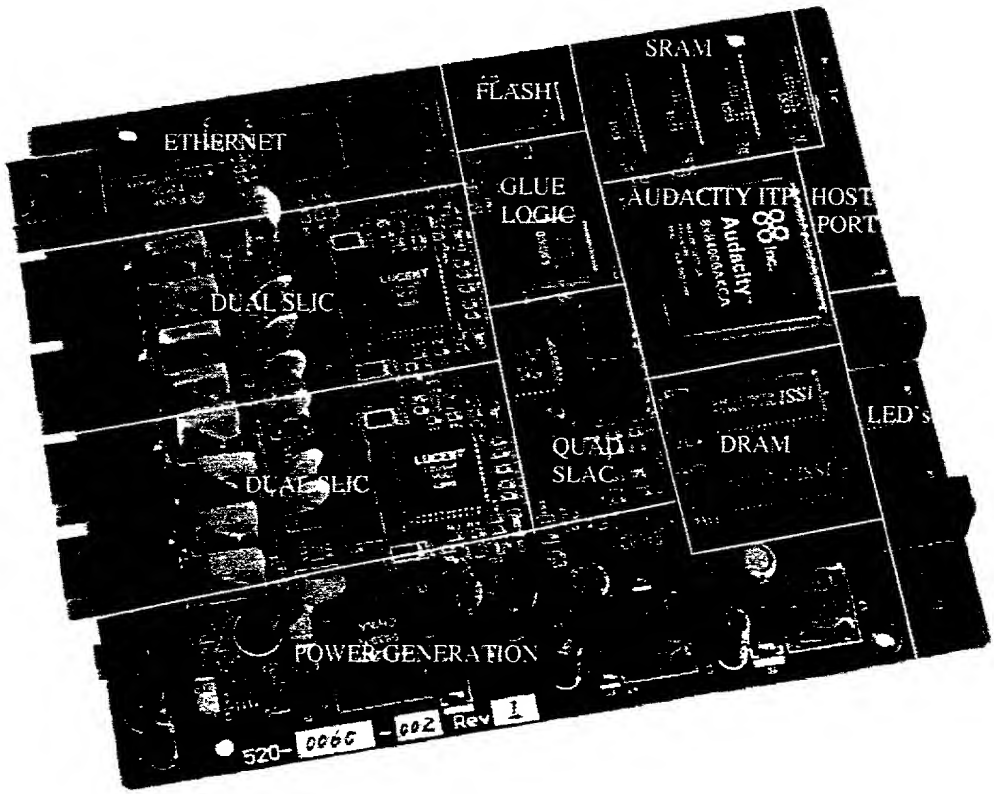


Figure 5: Symphony Mother board

Architecture

Symphony is a single board design, but consists of two major components: the network interface and the supporting motherboard functions. Figure 5 shows the general layout of the Symphony motherboard. The connectors (listed from left to right) are a RJ45 Ethernet jack, four RJ11 telephone jacks and a power connector.

Memory

The Symphony board uses three types of memory. The Audacity supports 4Mbits of FLASH EPROM, 512Kbytes of SRAM and 1 Mbyte of DRAM. Audacity is a pure 3.3V device all associated parts must be 3.3V tolerant.

SRAM

The SRAM is configured as 1 bank of 4 x 128kx8, 12ns asynchronous access and has a 32 bit wide data word.

DRAM

The DRAM is configured as 2 x 256kx16 of 50ns FPM or EDO memory and has a 32 bit wide data word.

FLASH
The Flash is configured as 4 Megabits, 512kx8 bits. For flexible erase and program capability, the 512kbits of data is divided into 11 sectors; one 16kbyte, two 8kbyte, one 32 kbyte, and seven 64 kbytes.

Glue logic

The Symphony glue-logic is based on a Lattice semiconductors core. Lattice supplies a suite of related tools allowing both design and in system programing of this part. The specific part used is an ISPLSI2128V-80LT100. This is a 3.3V I/O part and performs the following functions.

- TDM clock generation
- TDM control Audacity & Codec
- SRAM mapped registers for SLIC control
- AC LED control
- SRAM mapped registers for LED control
- Ringing frequency generation SRAM mapped access to 10BaseT

The Lattice chip comes with glue-logic designed to suit the existing profile of the board. However, if the board has a different function or application to meet specific needs, new glue-logic may be needed to support the external hardware features from software. For a full description of the lattice design, please refer to tables 1 - 3 & Appendix 2 – Glue-logic.

Ethernet port

This port is used to pass data to and from the Symphony board to a local ethernet connection. The popular 10BaseT connector has been used and allows data to flow upto a maximum of 10Mbs.

Ethernet controller
The ethernet controller used in the design is an MB86964 from Fujitsu. This device supports the 10BaseT interface. The connection to pass data to Audacity uses the SRAM port. However as the Ethernet controller is 5V and the Audacity is 3.3V, a voltage transceiver is needed between the two. The signals buffered are the LSB 16 bits of the SRAM port. Two control signals are used to control the flow of data - DIRECTION CONTROL & OUTPUT ENABLE. Both of these are generated from the glue-logic which is controlled by Audacity.

Ethernet memory
The ethernet controller requires a single 32k X 8 SRAM for local scratch memory.

Power

+5V Generation
There are a number of parts in the circuit which require 5V supply: T5504/7504, voltage transceivers, MB86964, 32kx8 SRAM, L8576, L8576. The power supply circuit is based on a LM2671M-5 switcher which derives its power from the main input power to the board.

+3V Generation
The majority of the board is designed around a 3.3V power source. These are the significant parts of the circuit which require 3.3V supply: Audacity, voltage transceivers, 2128V, 128kx8 SRAM, 256kx16 DRAMS. The power supply circuit is based on an LM2675M-3.3 switcher which derives its power from the main input power to the board.

Battery Voltage Generation
The design of the battery power supply is based on a Unitrode UC2572D which produces the constant -65V dc needed to power each line.

Power consumption
Maximum power consumption is not directly related to ringing, but to the number of phones off-hook. When all four lines are off-hook, maximum power consumption is reached. Total power consumption is 13W with an 18V supply. The 3.3V line supplies 3W and the 5V line supplies 2.5W. Each SLIC has a 1W drop-resistor, with each channel dissipating 1.8W. For lifeline and low power situations three of the four SLIC channels can be disabled, reducing SLIC power consumption to 0.2W. Full system power down in sleep mode will be 0.5W. Table 4 shows the power consumption for each sub-system on the board.

Address	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Description
0x0000000 – 0x007FFFF	R/W									Base SRAM (512K Byte)
0x8000000 – 0x800001C	R/W									Ethernet Controller (Word Access on LongWord Boundary)
0x8080000	R/W	X	X	X	STATUS0	SLIC_LED0		RING0	BP0	SLIC 0 CONTROL
0x8080004	R/W	X	X	X	STATUS1	SLIC_LED1		RING1	BP1	SLIC 1 CONTROL
0x8080008	R/W	X	X	X	STATUS2	SLIC_LED2		RING2	BP2	SLIC 2 CONTROL
0x808000C	R/W	X	X	X	STATUS3	SLIC_LED3		RING3	BP3	SLIC 3 CONTROL
0x8080010	R/W	E_IRQ	X	X	SLICS_ON	TDM_512	ALAW	E_RESET	E_IRQ_EN	SYSTEM CONTROL
0x8080014	R/W	LINK_LED3		LINK_LED2		LINK_LED1		LINK_LED0		LINK LED CONTROLS
0x80FFFFF	R	SYSID7	SYSID6	SYSID5	SYSID4	SYSID3	SYSID2	SYSID1	SYSID0	SYSTEM ID

Table 1: Lattice memory map

Signal Name	Description
E_IRQ	Current Interrupt Request status of MB86964
E_IRQ_EN	Connect E_IRQ to Audacity
E_RESET	Set to RESET MB86964 Reset to enable MB86964
SLICS_ON	Set to enable SLICS 1-3 Reset to lower system power
TDM_512	Set TDM = 512bpf, 64*8 slots Reset TDM 256 bpf, 32*8 slots
ALAW	Set for A-Law, 8 bit data Reset for u-Law, 8 bit data

Table 2: Ethernet Glue-logic controls

Ring (Input)	Battery Polarity (Input)	Line State	SLIC_STATUS (Output)
0	0	Reverse Battery Feed	Loop Current
0	1	Forward Battery Feed	Loop Current
1	X	20Hz Ring	Ring Trip

Table 3: SLIC controls

Color	Active	Sleep	Notes
Audacity IC	1000	300	
SRAM	1500	100	
DRAM	800	75	
Lattice	300	30	estimated
Oscillators	180	180	
LEDs	250	0	
Ethernet	1100	50	
Codec	350	0	
SLICs	8000	150 275	suspend polling

Table 4: Power consumption (mA)

Color	Steady	Slow Blink	Fast Blink
Off	Reset	---	---
Red	---	---	---
Yellow	Off hook	---	Ringing
Green	Ready	Message	---

Table 5: Line 1-4 LED functions

Color	Steady	Slow Blink	Fast Blink
Off	Reset	---	---
Red	Initializing	---	---
Yellow	---	---	---
Green	Configured	---	---

Table 6: Call Agent ready LED

Color	Steady	Slow Blink	Fast Blink
Off	Reset	---	---
Red	Initializing TCP/IP stack	---	---
Yellow	Acquiring IP Address	---	Data Read
Green	Initialized	---	---

Table 7: MGCP link status LED

Color	Steady	Slow Blink	Fast Blink
Off	Reset	---	---
Red	MGCP Engine Initialized	---	Fault
Yellow	Audio Subsystem Initialized	---	Ringing
Green	Initialized and ready	---	---

Table 8: System Status LED

TDM data protocol

The TDM port implements a 5 wire serial bus which provides an easy connection between the VCP and available communication chips. The TDM bus is a Time Division Bus multiplexing data on up to 64 channels. Each channel is allocated a different time slot on the bus, and the VCP can be set to different time slots. Data is assumed to be ordered by time slot (i.e. if timeslots 6,8 and 17 are used, the first byte DMA'ed to the memory is the byte in time slot 6, followed by 8 and 17 in order. Reordering must be done in software. The interface consists of a frame-sync (TDMFS), data transmit and receive signals (TDMDX, TDMDR), external buffer enable (TDMTSC#) and a clock (TDMCLK). The frame sync and Clock signals are inputs to the VCP, so the timing of the data transfer is externally controlled. The TDM port can support a number of different timings on the TDM bus. The TDM bus runs from the Audacity ITP to the Quad PCM CODEC. The TDM interface to the codec must be at 8k sample rates, 256/512k bits per frame. Time slots used are 0,1,2,3; other time slots can be used for different applications.

LEDs

A number of LEDs are provided for system status feedback. The LEDs are arranged in a dual LED arrangement. Each LED pair can convey 7 states – Off, Red, Yellow, Green, Steady, Slow Blink and Fast Blink.

Ready, Link & Status are conveyed in three LED pairs. The SLIC status of Lines 1-4 are conveyed in four LED pairs. Tables 5-8 show the mapping. One Further LED pair exists for future system expansion.

Connectors

This section covers all connectors on the motherboard, for more detailed information on the connectors refer to the Symphony design schematics and glue-logic equations.

J8 – RJ45 connector.
This connector accepts a 10BaseT style ethernet connection for passing data to and from an ethernet network.

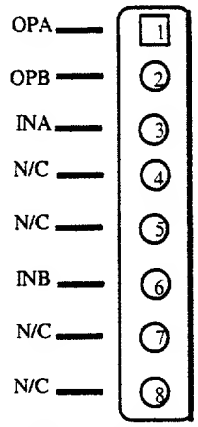


Figure 7: Ethernet connector (J8)

J2,4,5,6 – RJ11 connectors.
These connectors are used to connect external telephony equipment, such as Telephones and Fax machines to the Symphony board. There are 4 RJ11 connectors which can be used for equipment connection.

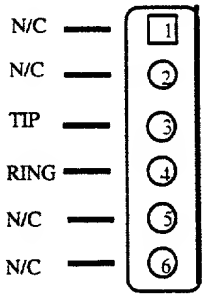


Figure 8: Line Connectors (J2, J4, J5, J6)

J1 - Power supply connector.
This is where the main power supply for the Symphony motherboard enters. An unregulated +18V is produced by the external power supply and enters the board via this connector. Positive is the center pin.

J3 – First TDM port connector.
The TDM port connector carries signals for the transfer of fast streams of data to and from the Audacity TDM port. It also carries TDM clock and frame syncs provided by the attached subsystem. However, there are two TDM connectors on Symphony in parallel with each other. This is the primary connector. Applications requiring TDM data transfer are typically found in many types of add-in communication card.

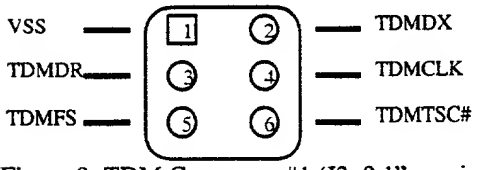


Figure 9: TDM Connector #1 (J3, 0.1" spacing)

J9 – Second TDM port connector.
This port is the secondary TDM connector. It is physically in line with the 3.3V host port connector for applications which require host and TDM data transfer together. Although it has an identical pin-out as the primary TDM connector, a single 50 way connector can be placed upon the secondary TDM port and host port connectors to achieve this connectivity feature. The pin out is shown below in figure 10.

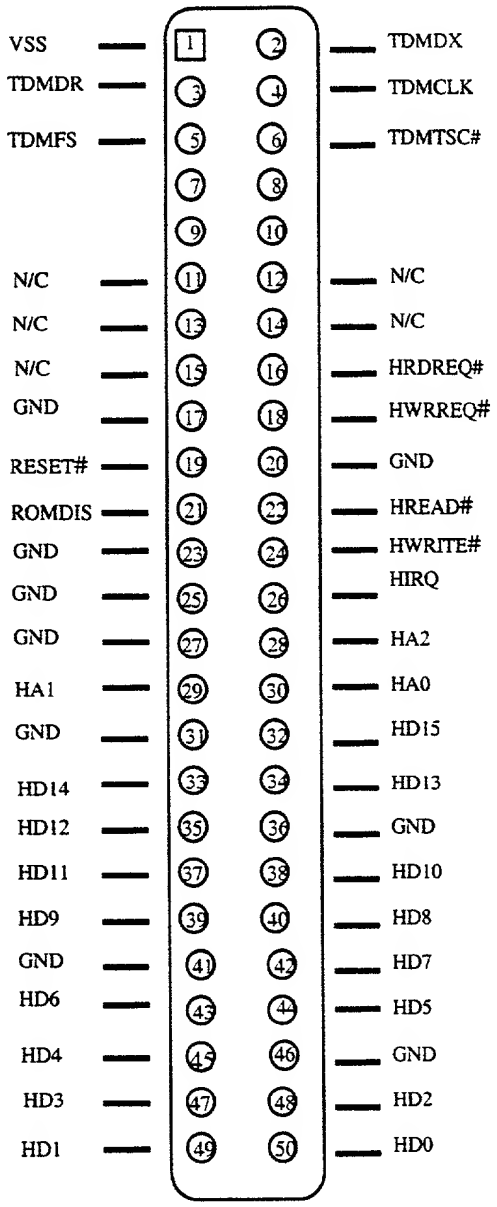


Figure 10: Host & TDM Connector

J10 – Lattice header.
This header is used to in circuit program the Lattice glue-logic IC. The design and download software for this application are available from Lattice semiconductor directly.

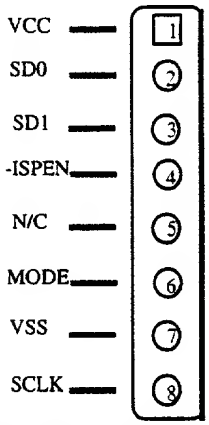


Figure 11: Lattice Programming connector (J10)

J7 – 3.3V Host port connector.
This is the connector used to connect to a 3.3V STIC card. This connector is used for debugging and for downloading programs when not booting from onboard ROM this connector is located at the edge of the board for easy access.

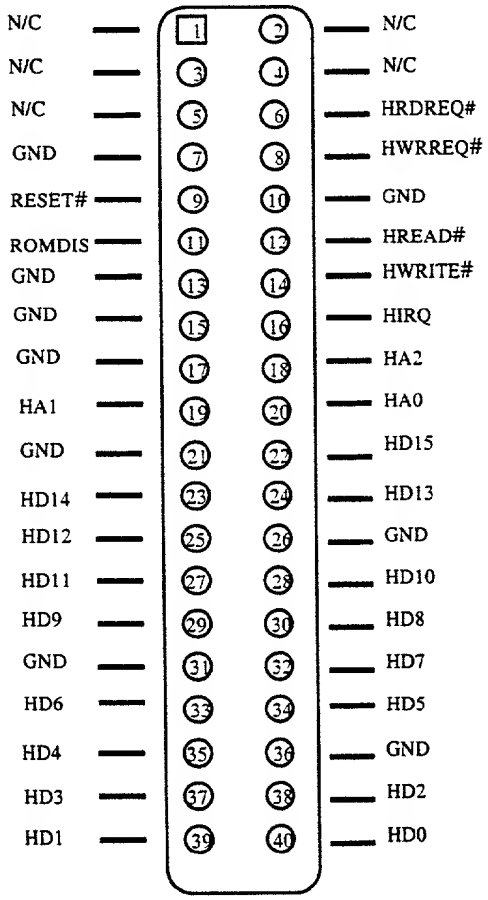
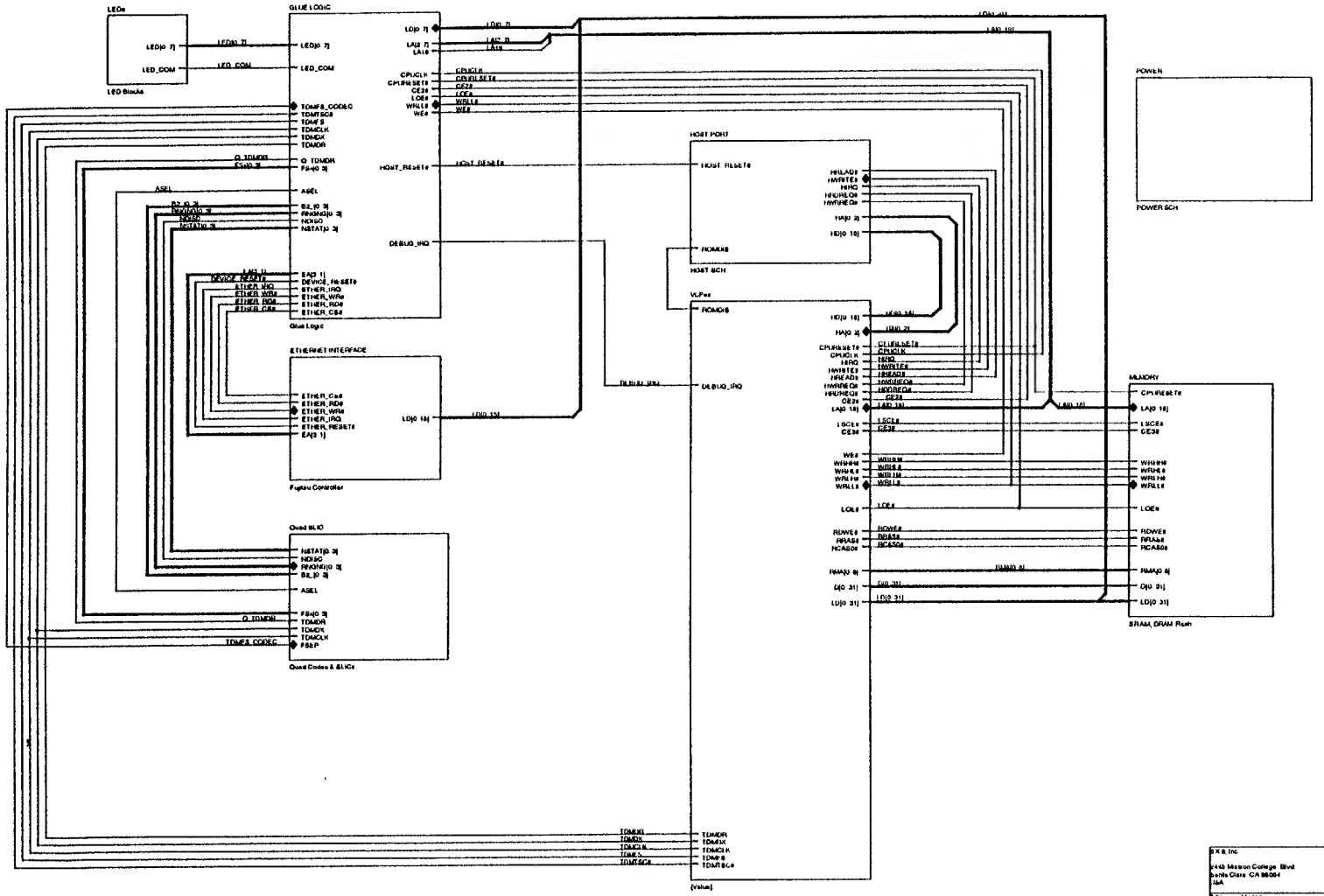


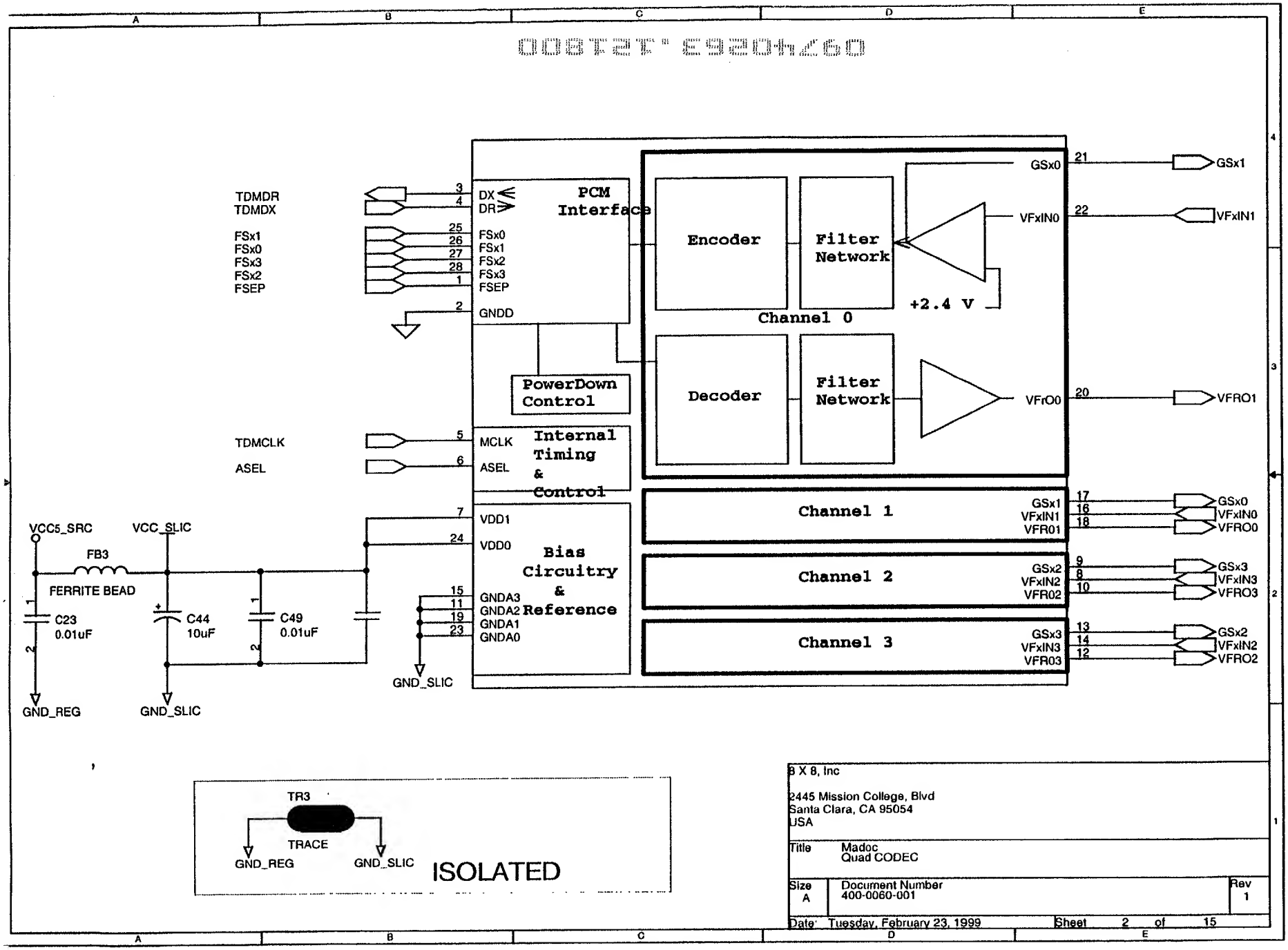
Figure 12: 3.3V Host Debug Connector (J7)

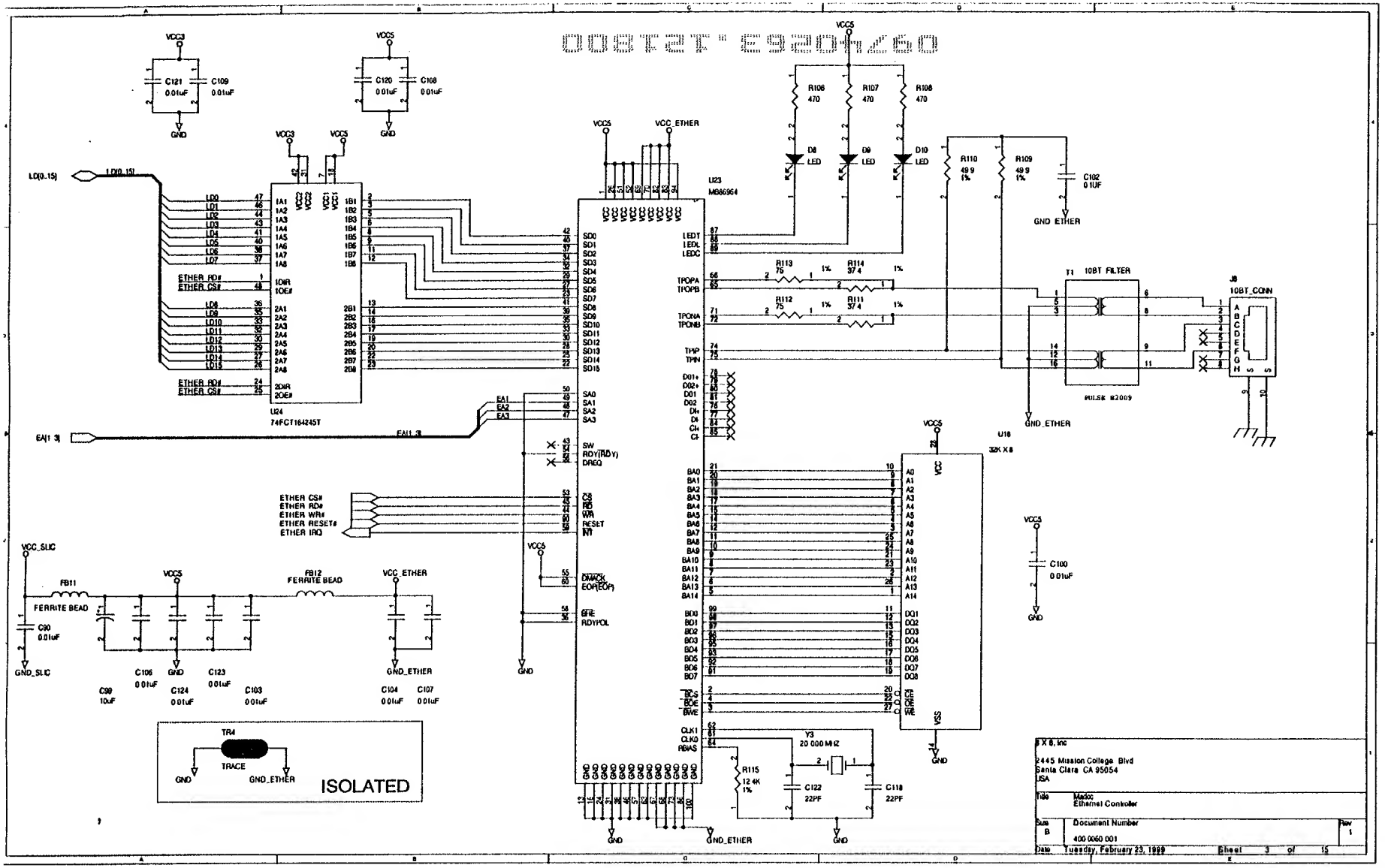
Appendix 1 – Schematics

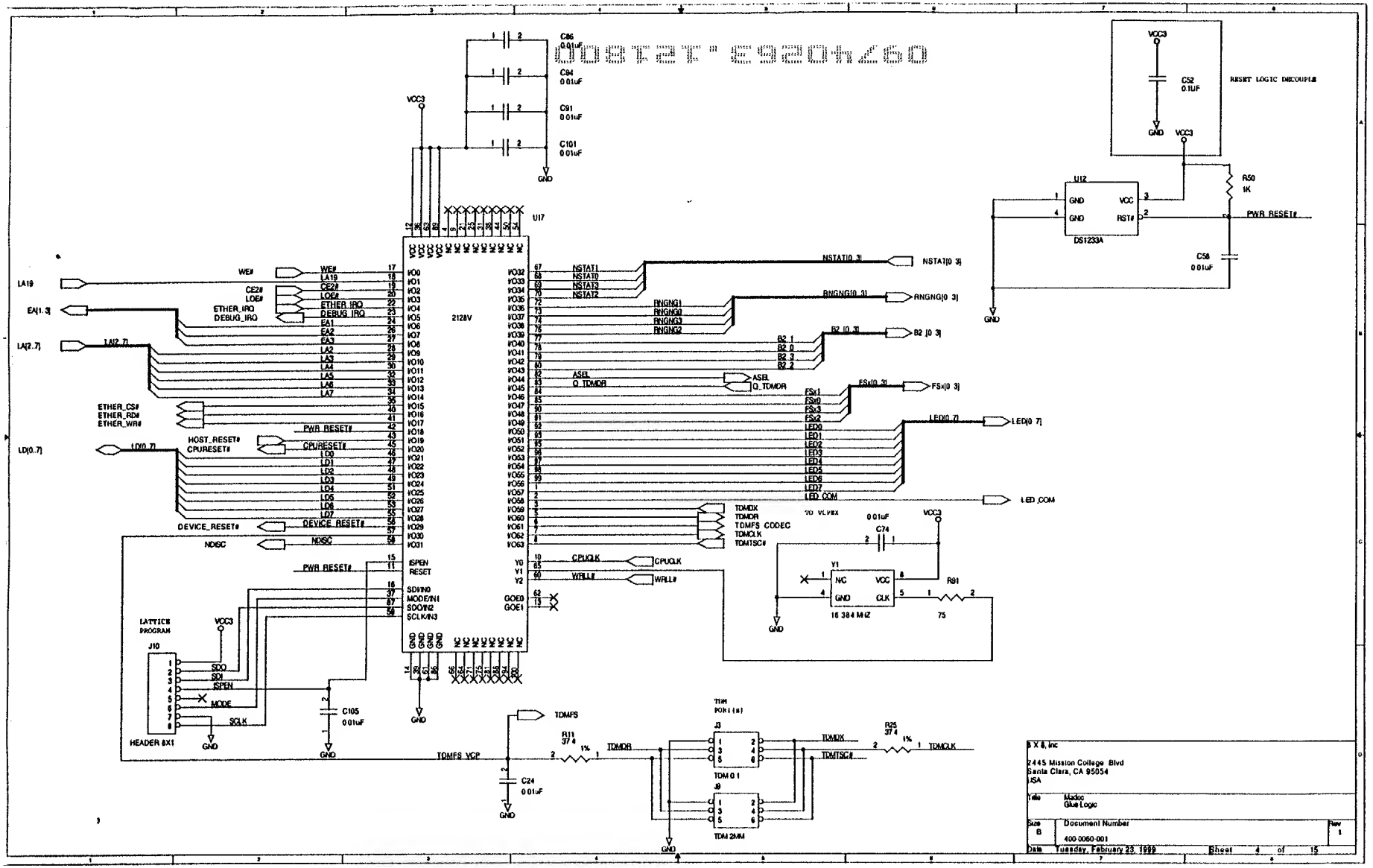
Confidential

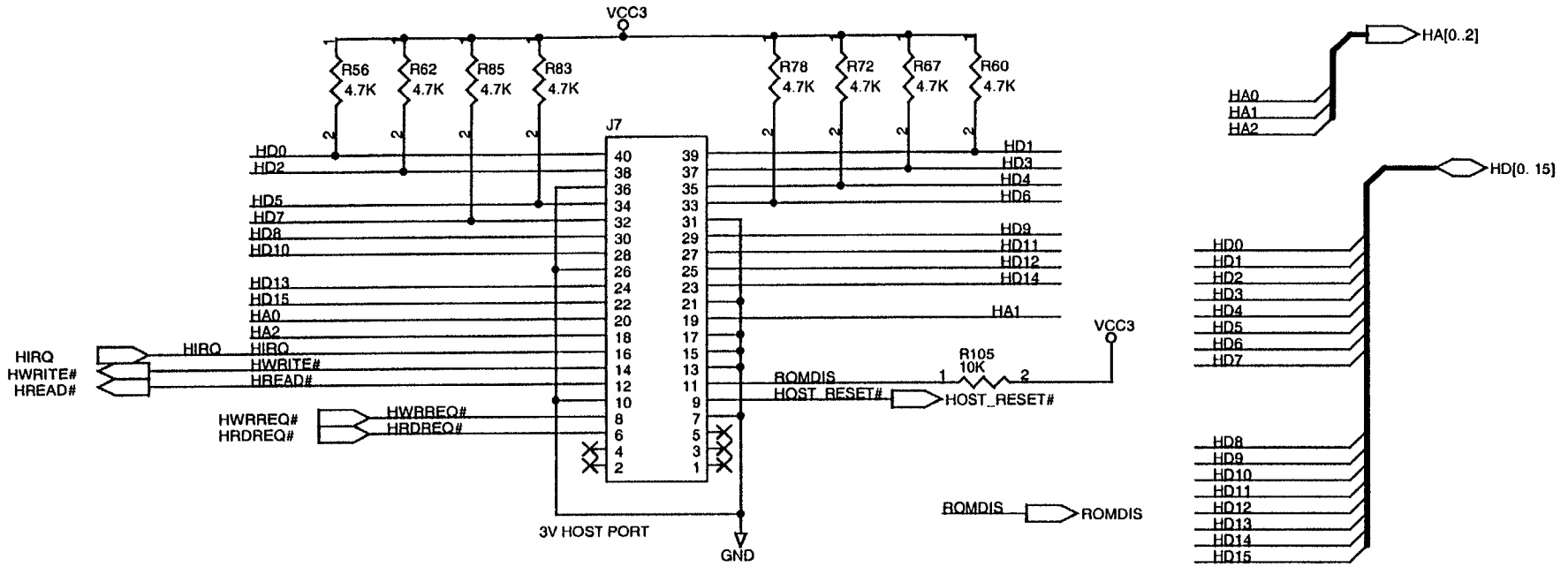


BXB, Inc		
3448 Mission College Blvd Santa Clara CA 95054 USA		
Title	Module Hierarchy	
Size: C	Document Number: 400-0060-001	Rev: 1
Date: Tuesday, February 23, 1999	Sheet 1 of 18	


PCM Interface
DX
DR
FSx0
FSx1
FSx2
FSx3
FSEP
GNDD
TDMDR
TDMDX
FSx1
FSx0
FSx3
FSx2
FSEP
PowerDown Control
Internal Timing & Control
MCLK
ASEL
TDMCLK
ASEL
Bias Circuitry & Reference
VDD1
VDD0
GNDA3
GNDA2
GNDA1
GNDA0
Encoder
Filter Network
Decoder
Filter Network
+2.4 V
Channel 0
Channel 1
Channel 2
Channel 3
GSx0
VFxIN0
VFrO0
GSx1
VFxIN1
VFR01
GSx2
VFxIN2
VFR02
GSx3
VFxIN3
VFR03
GSx1
VFxIN1
VFRO1
GSx0
VFxIN0
VFRO0
GSx3
VFxIN3
VFRO3
GSx2
VFxIN2
VFRO2
VCC5_SRC
VCC_SLIC
FB3
FERRITE BEAD
C23
0.01uF
C44
10uF
C49
0.01uF
GND_REG
GND_SLIC
GND_SLIC
TR3
TRACE
GND_REG
GND_SLIC
ISOLATED
8 X 8, Inc
2445 Mission College, Blvd
Santa Clara, CA 95054
USA
Title
Madoc
Quad CODEC
Size
A
Document Number
400-0060-001
Rev
1
Date: Tuesday, February 23, 1999
Sheet 2 of 15


C121 0.01uF
C109 0.01uF
C120 0.01uF
C108 0.01uF
R106 470
R107 470
R108 470
D8 LED
D9 LED
D10 LED
R110 49.9 1%
R109 49.9 1%
C102 0.1UF
GND_ETHER
LD(0..15)
U24
74FCT164245T
U23
MB86964
R113 75 1%
R114 374 1%
R112 75 1%
R111 374 1%
T1 10BT FILTER
J8 10BT_CONN
EA(1..3)
ETHER CS#
ETHER RD#
ETHER WR#
ETHER RESET#
ETHER IRQ
U18
32K X 8
C100 0.01uF
VCC_SLIC
FB11
FERRITE BEAD
FB12
FERRITE BEAD
C90 0.01uF
C106 0.01uF
C123 0.01uF
C99 10uF
C124 0.01uF
C103 0.01uF
C104 0.01uF
C107 0.01uF
GND_SLIC
TR4
TRACE
ISOLATED
R115 12.4K 1%
Y3 20.000 MHz
C122 22PF
C118 22PF
8 X 8, Inc
2445 Mission College Blvd
Santa Clara CA 95054
USA
Title
Master Ethernet Controller
Size B
Document Number
400 0060 001
Rev 1
Date Tuesday, February 23, 1999
Sheet 3 of 15


C86 0.01uF
C94 0.01uF
C91 0.01uF
C101 0.01uF
VCC3
GND
U17
2128V
WE#
LA19
CE2#
LOE#
ETHER_IRQ
DEBUG_IRQ
EA[1..3]
LA[2..7]
ETHER_CS#
ETHER_RD#
ETHER_WR#
PWR_RESET#
HOST_RESET#
CPURESET#
LD[0..7]
DEVICE_RESET#
NDISC
ISPEN
RESET
SDI/IN0
MODE/IN1
SDO/IN2
SCLK/IN3
GOE0
GOE1
NSTAT[0..3]
RNGNG[0..3]
B2 [0..3]
ASEL
Q_TDMDR
FSx[0..3]
LED[0..7]
LED_COM
TDMDX
TDMDR
TDMFS_CODEC
TDMCLK
TDMTSC#
CPUCLK
WRLL#
LATTICE PROGRAM
J10
HEADER 8X1
SDO
SDI
ISPEN
MODE
SCLK
C105 0.01uF
TDMFS_VCP
TDMFS
R11 374 1%
C24 0.01uF
TDMDR
J3
TDM 0 1
J9
TDM 2MM
R25 374 1%
TDMDX
TDMTSC#
TDMCLK
C74 0.01uF
Y1
16.384 MHZ
R91 75
C52 0.1UF
RESET LOGIC DECOUPLE
U12
DS1233A
R50 1K
PWR_RESET#
C58 0.01uF
B X 6, Inc
2445 Mission College Blvd
Santa Clara, CA 95054
USA
Title Madoc Glue Logic
Size B
Document Number 400-0060-001
Rev 1
Date Tuesday, February 23, 1999
Sheet 4 of 15

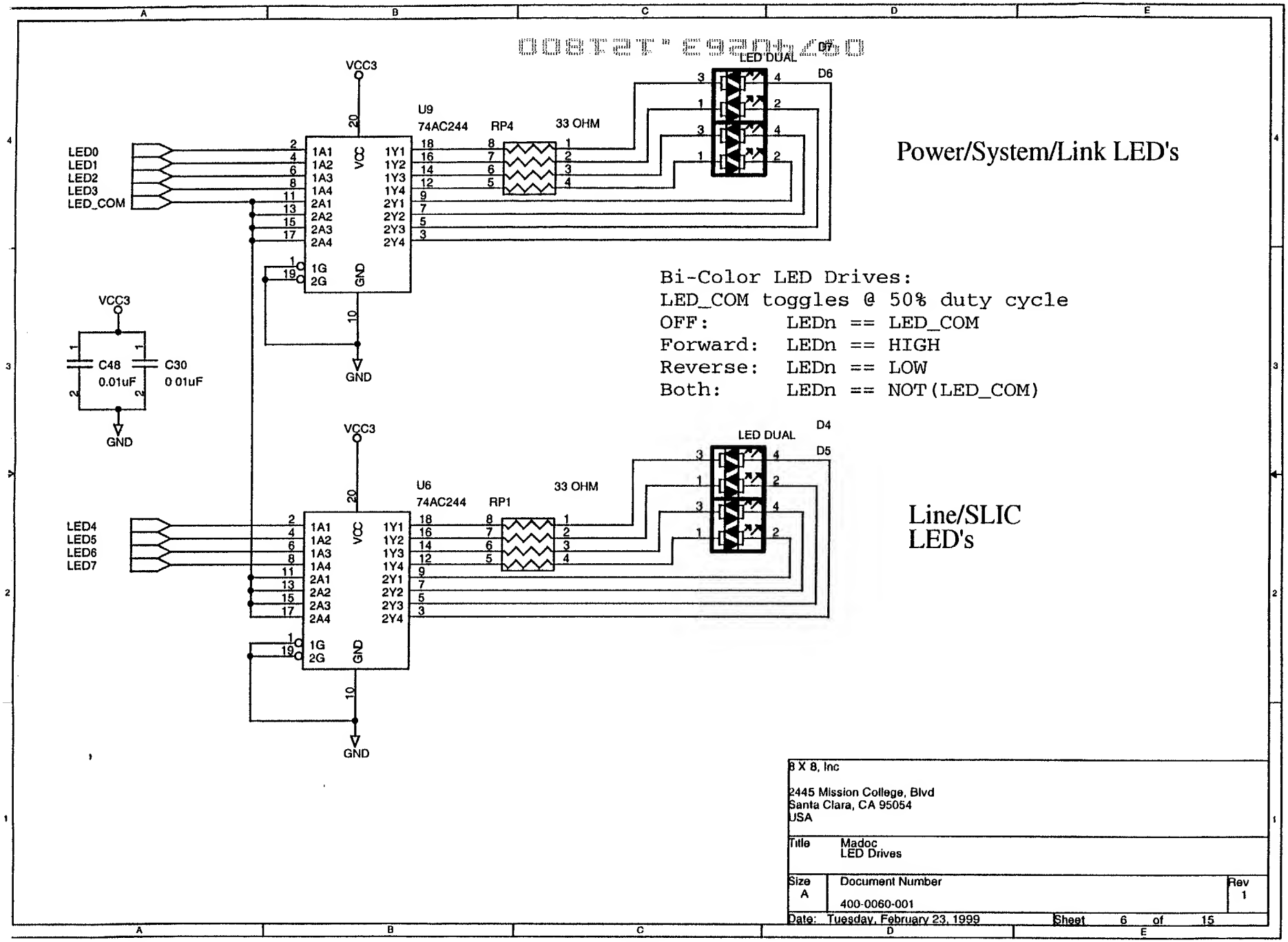


NOTES:
1. FOLLOW LAYOUT RULES FOR DIGITAL CIRCUITS.

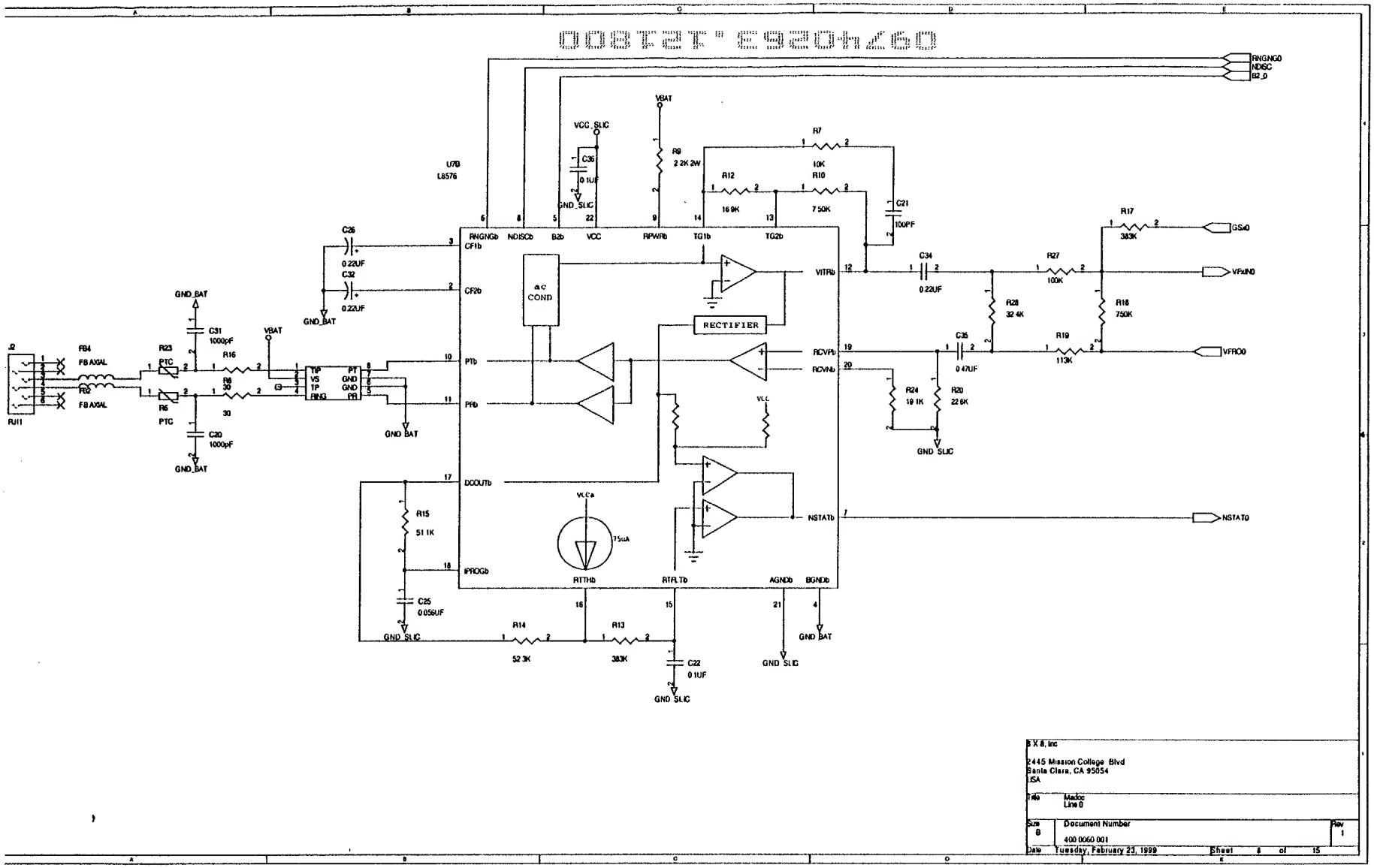
8 X 8, Inc

2445 Mission College, Blvd.
Santa Clara, CA 95054
USA

Title	Madoc Host Port	
Size A	Document Number 400-0060-001	Rev 1
Date: Tuesday, February 23, 1999	Sheet 5 of 15	



8 X 8, Inc 2445 Mission College, Blvd Santa Clara, CA 95054 USA		
Title	Madoc LED Drives	
Size A	Document Number 400-0060-001	Rev 1
Date: Tuesday, February 23, 1999	Sheet 6 of 15	

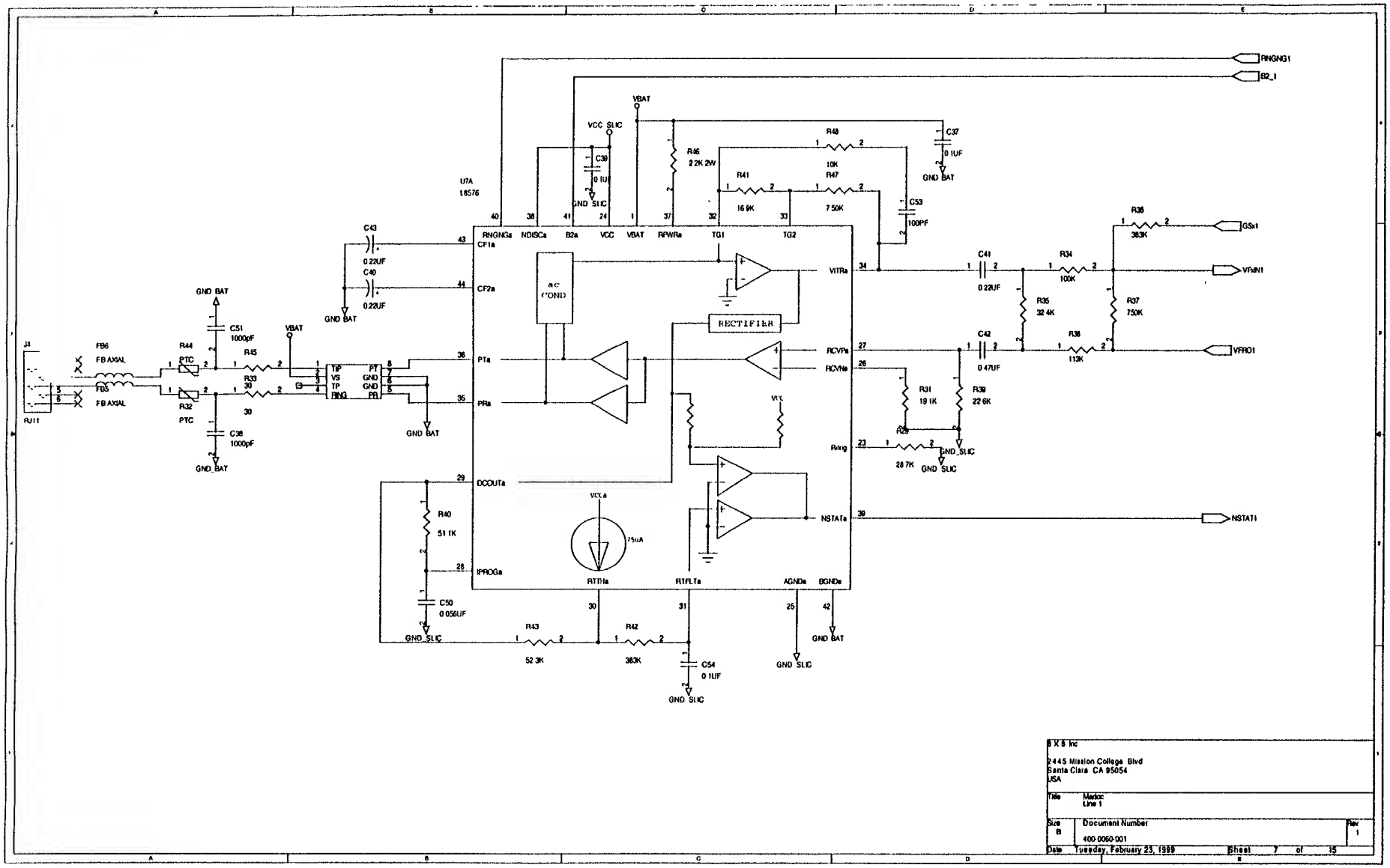


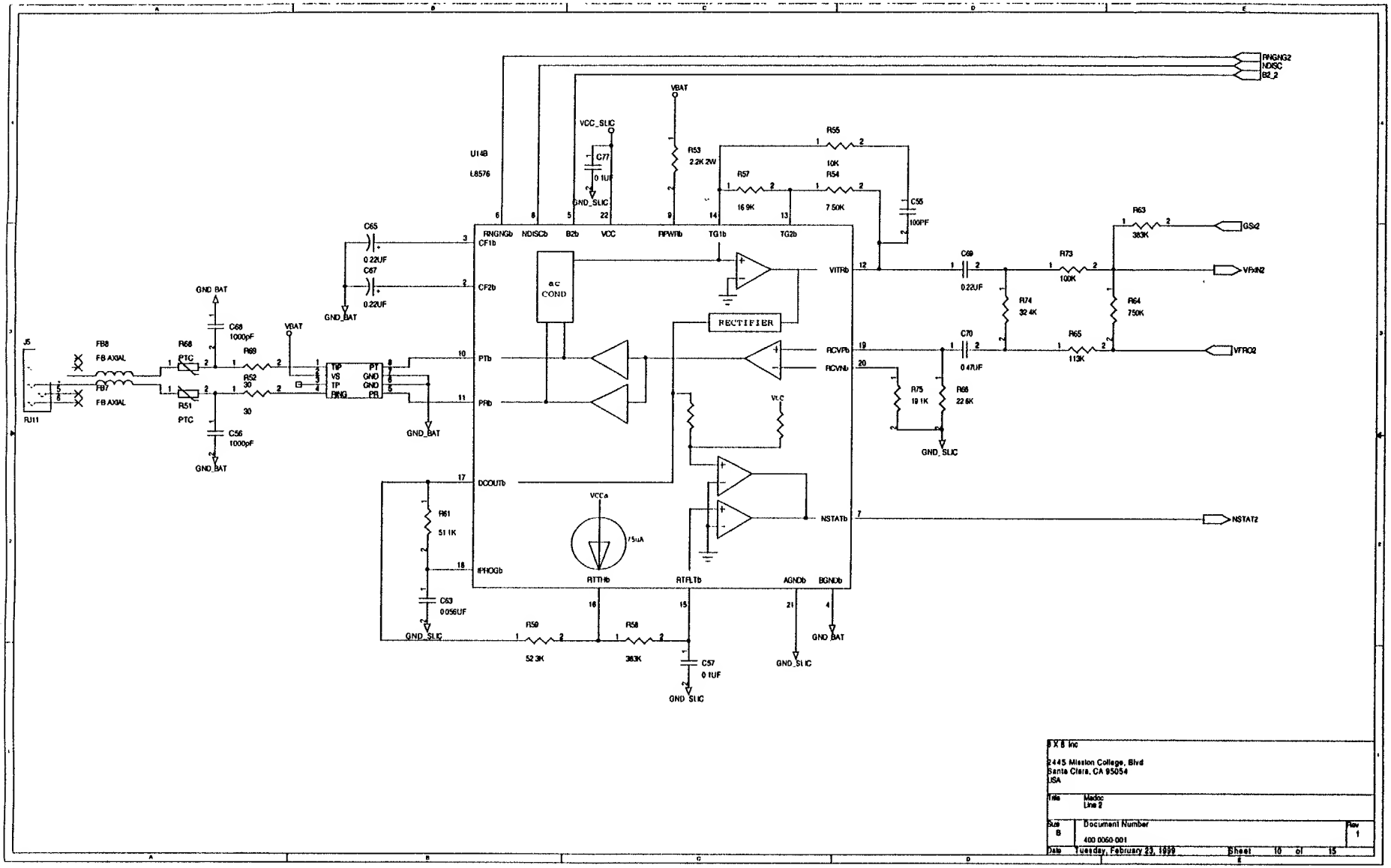
B X B, Inc
2445 Mission College Blvd
Santa Clara, CA 95054
USA

Title: Macloc
Line 0

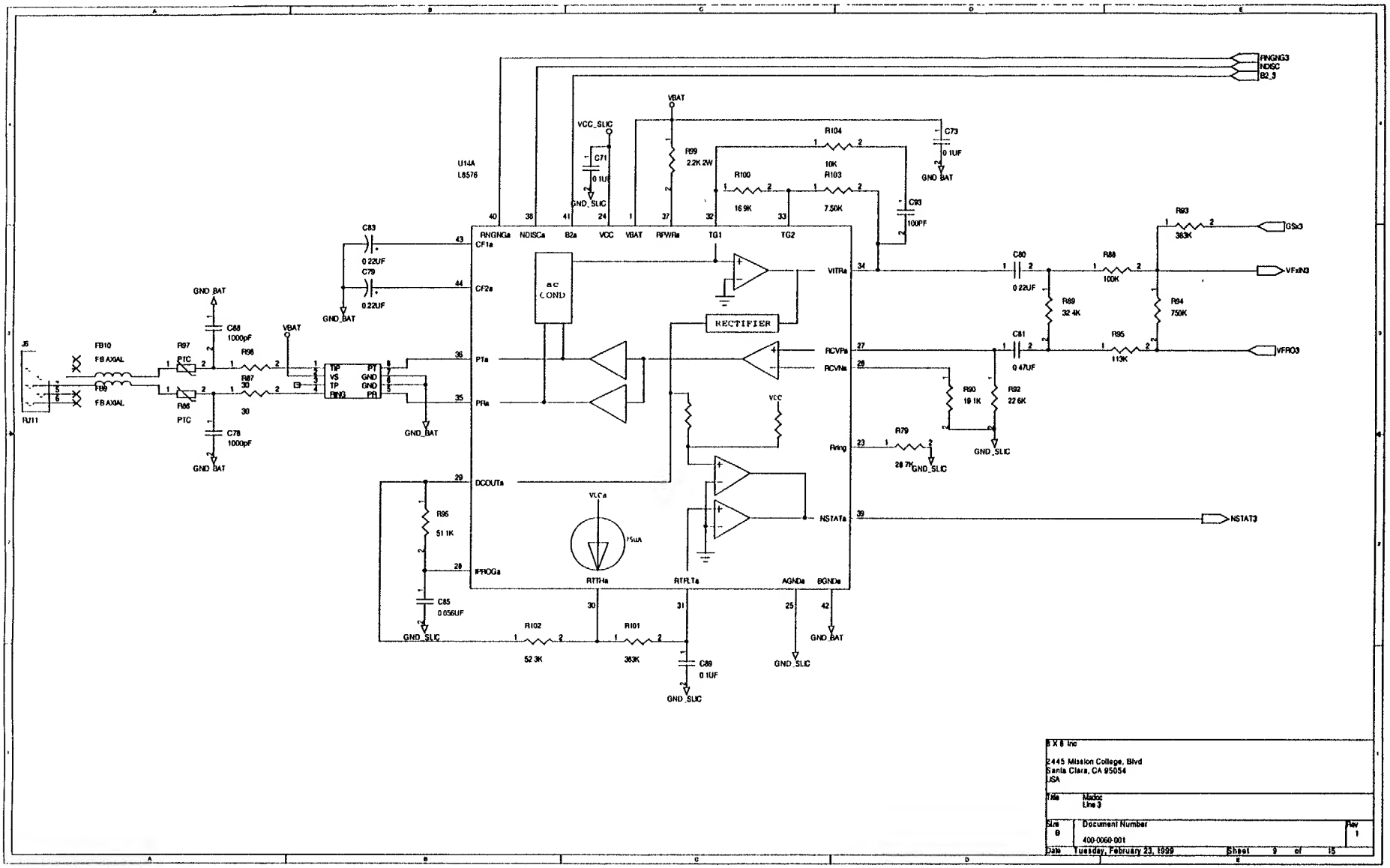
Size: B | Document Number: 400 0060 001 | Rev: 1

Date: Tuesday, February 23, 1999 | Sheet 8 of 15

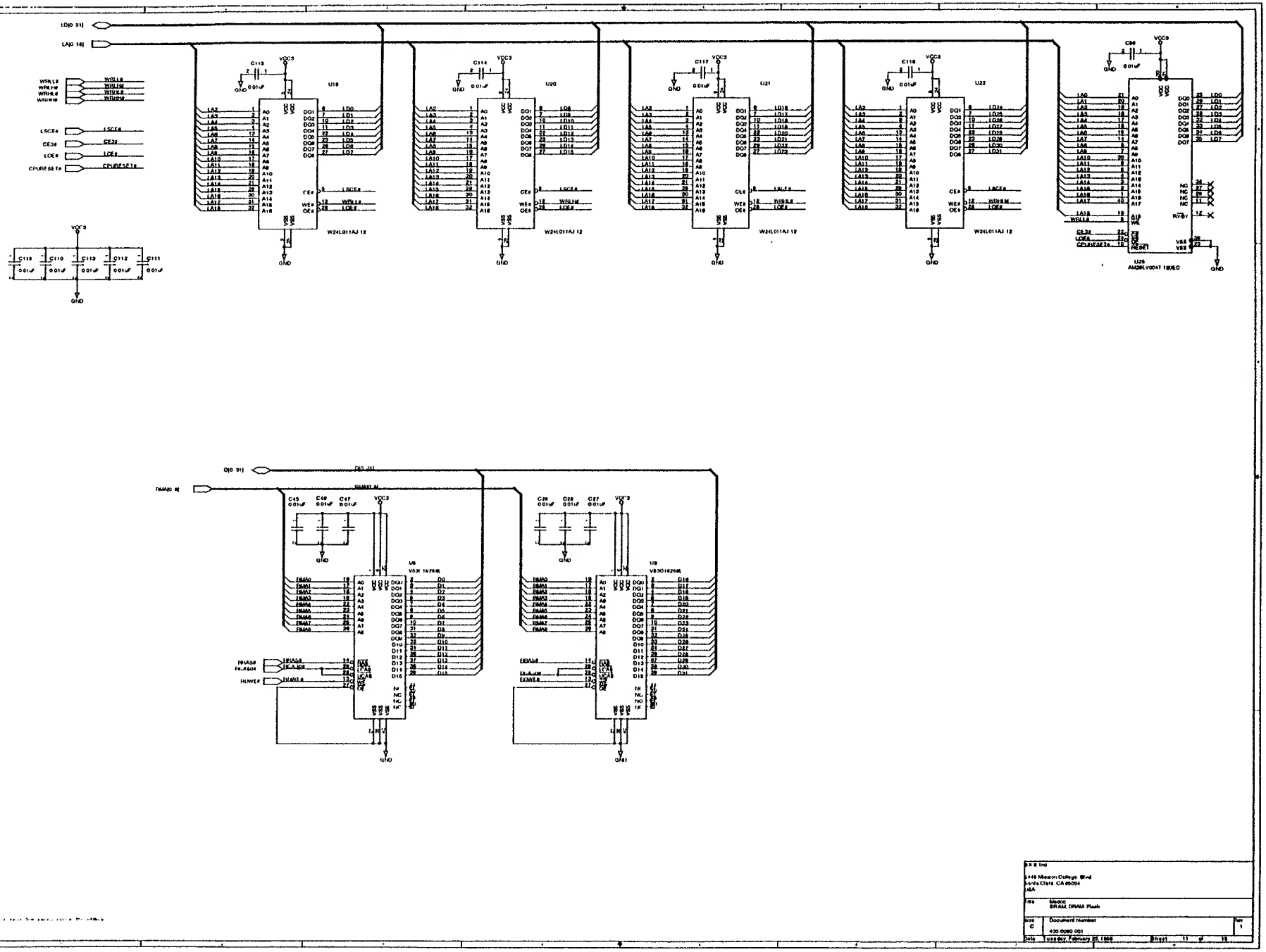

RNGNG1
B2_1
VBAT
VCC_SLIC
U7A
L8576
C37
0.1UF
GND_BAT
R48
10K
R46
2.2K 2W
R41
16.9K
R47
7.50K
C38
0.1UF
GND_SLIC
C53
100PF
R36
383K
GSx1
C43
0.22UF
C40
0.22UF
RNGNGa
CF1a
CF2a
NDISCa
B2a
VCC
VBAT
RPWRa
TG1
TG2
VITRa
C41
0.22UF
R34
100K
VFxIN1
R35
32.4K
R37
750K
R38
113K
VFRO1
RECTIFIER
RCVPa
RCVNa
C42
0.47UF
R31
19.1K
R39
22.6K
Rring
28.7K
GND_SLIC
J1
FB6
FB AXIAL
FB5
FB AXIAL
RJ11
R44
PTC
R45
C51
1000pF
R33
30
R32
PTC
C36
1000pF
GND_BAT
TIP
VSI
TIP
RING
PT
GND
GND
PR
PTa
PRa
DCOUTa
R40
51.1K
C50
0.056UF
IPROGa
VCCa
75uA
RTIPa
RTFLTa
AGNDa
BGNDa
NSTATa
NSTAT1
R43
52.3K
R42
383K
C54
0.1UF
GND_SLIC
S.X.8 Inc
2445 Mission College Blvd
Santa Clara CA 95054
USA
Title
Mediac
Line 1
Size
B
Document Number
400-0060-001
Rev
1
Date
Tuesday, February 23, 1999
Sheet
7
of
15



8 X 8 Inc 2445 Mission College, Blvd Santa Clara, CA 95054 USA		
Title	Madoc Line 2	
Size B	Document Number 400 0060 001	Rev 1
Date	Tuesday, February 23, 1999	Sheet 10 of 15

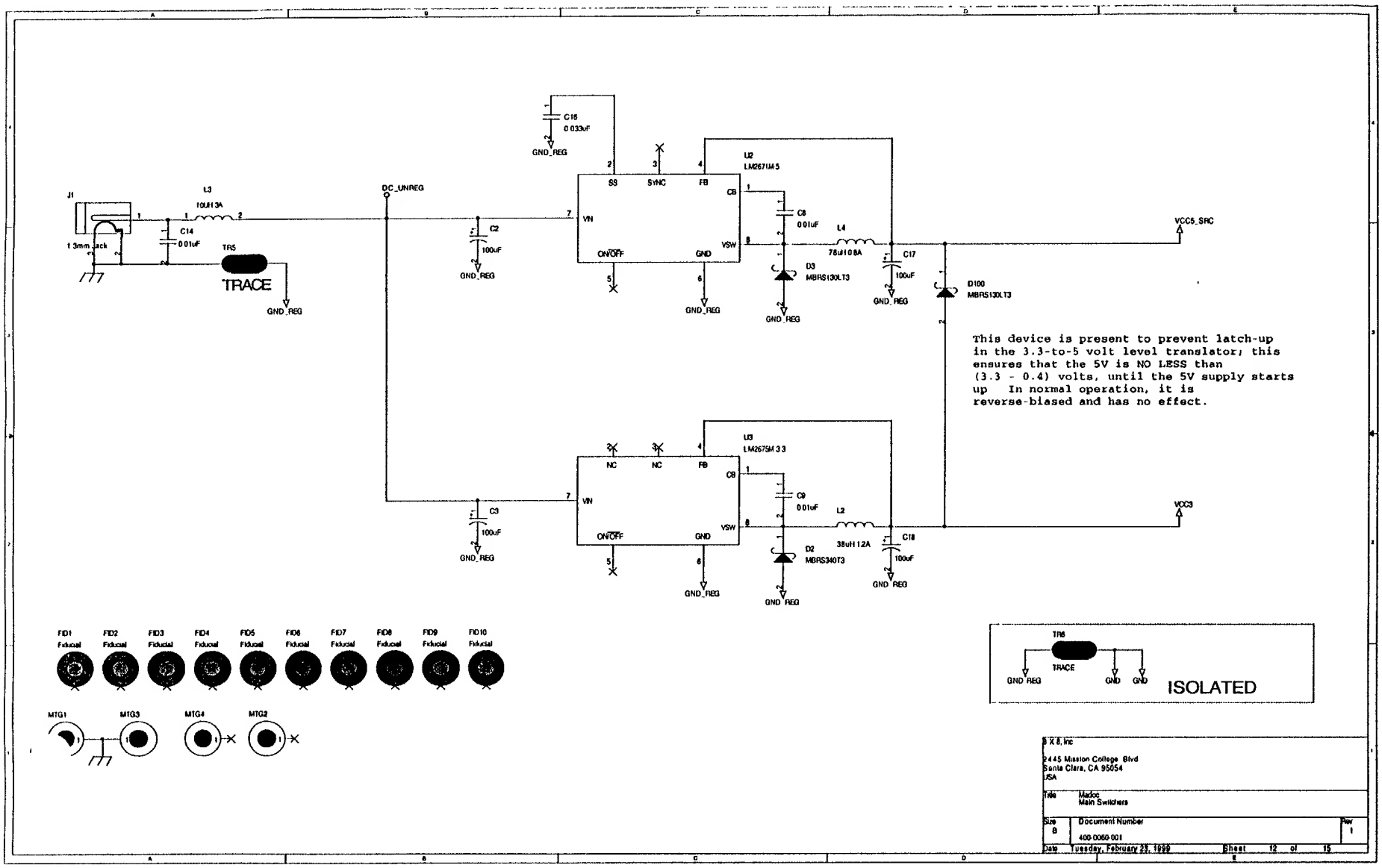


09/40263 .121800

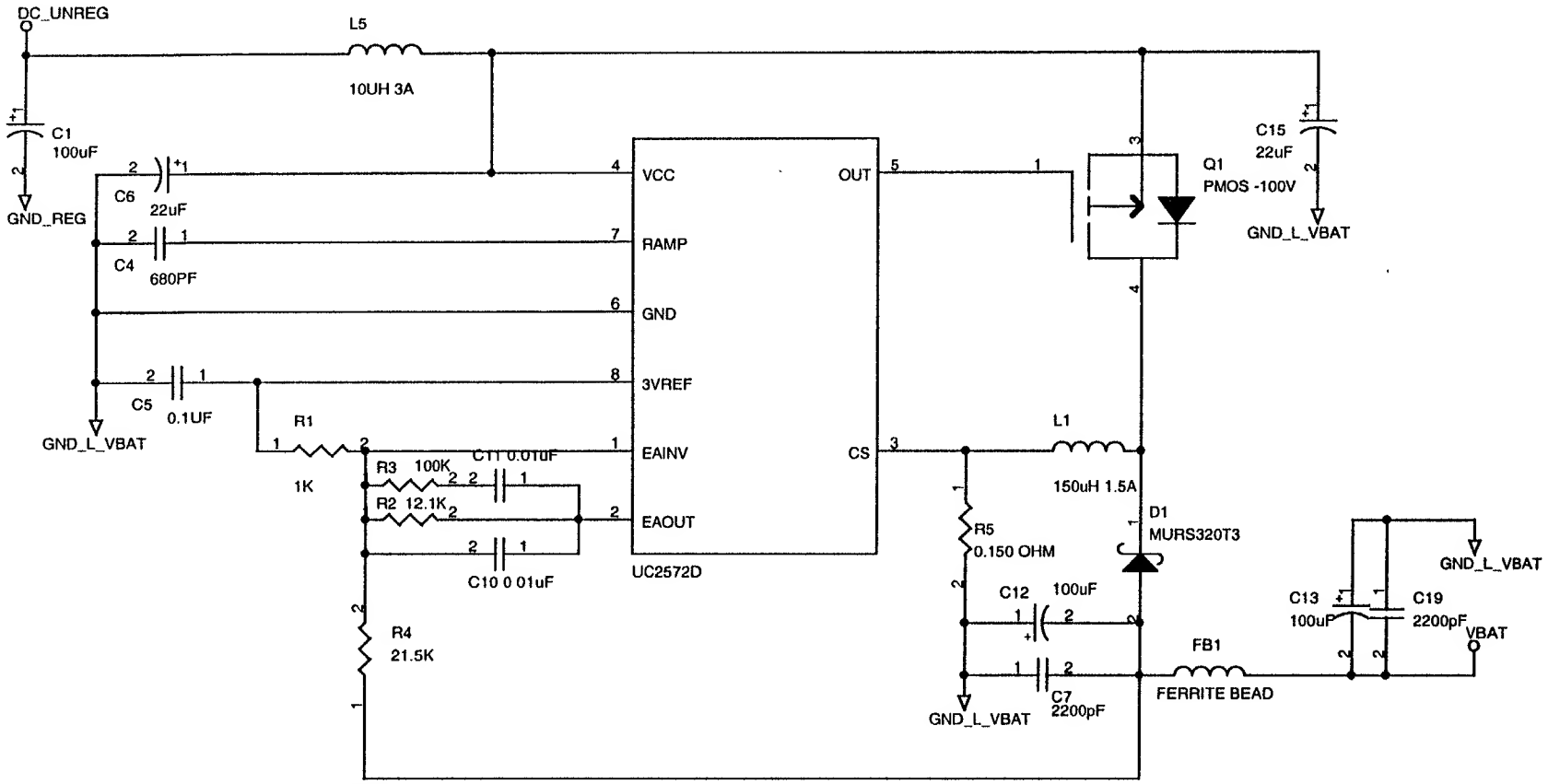


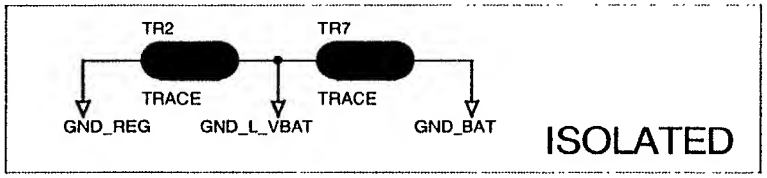
Title	Module SRAM, DRAM, Flash	
Size C	Document Number 400 0080 001	Rev 1
Date	Tuesday, February 25, 1998	Sheet 11 of 19

Document Control - Restricted



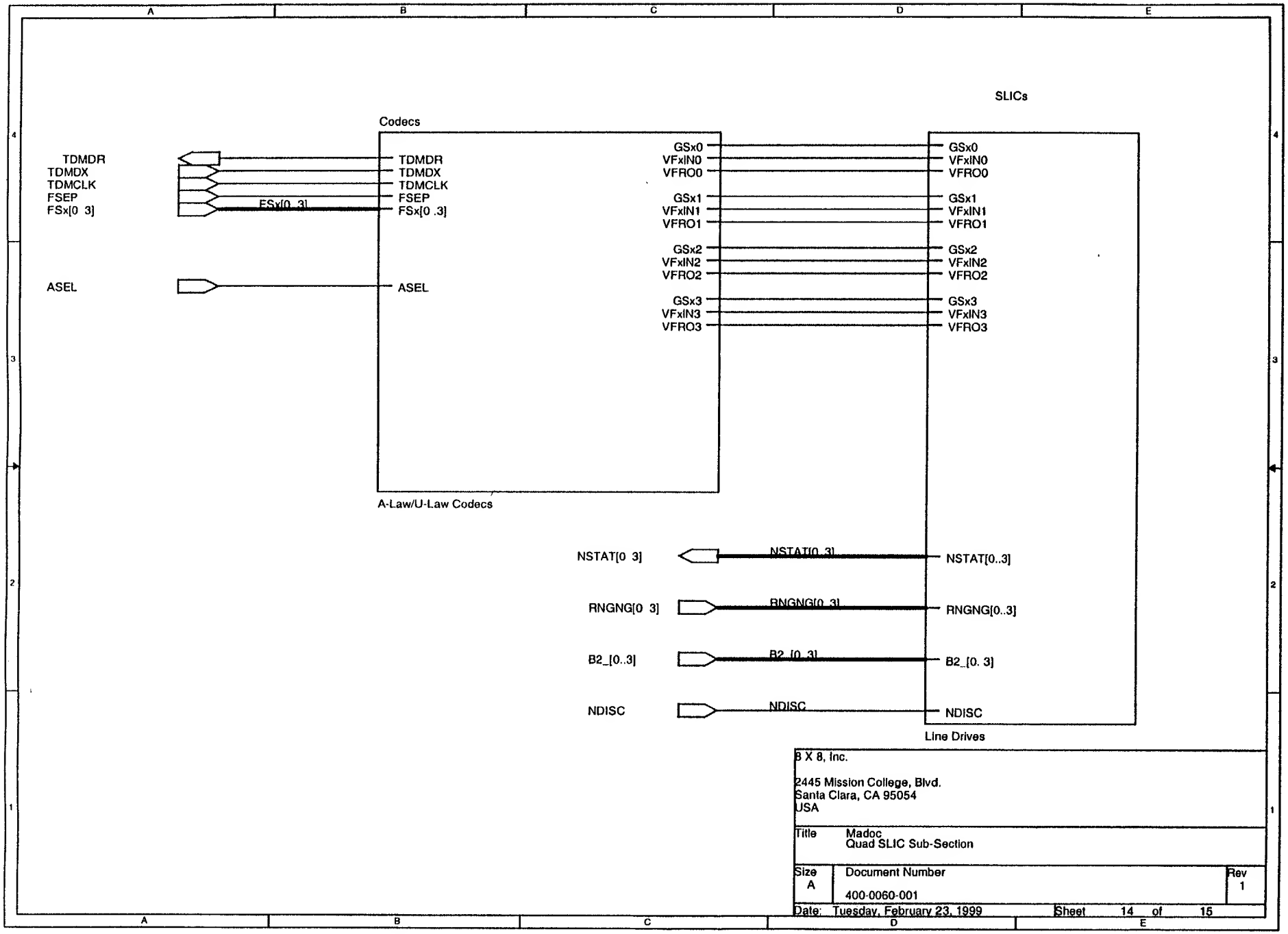
D3240E3 .12180D



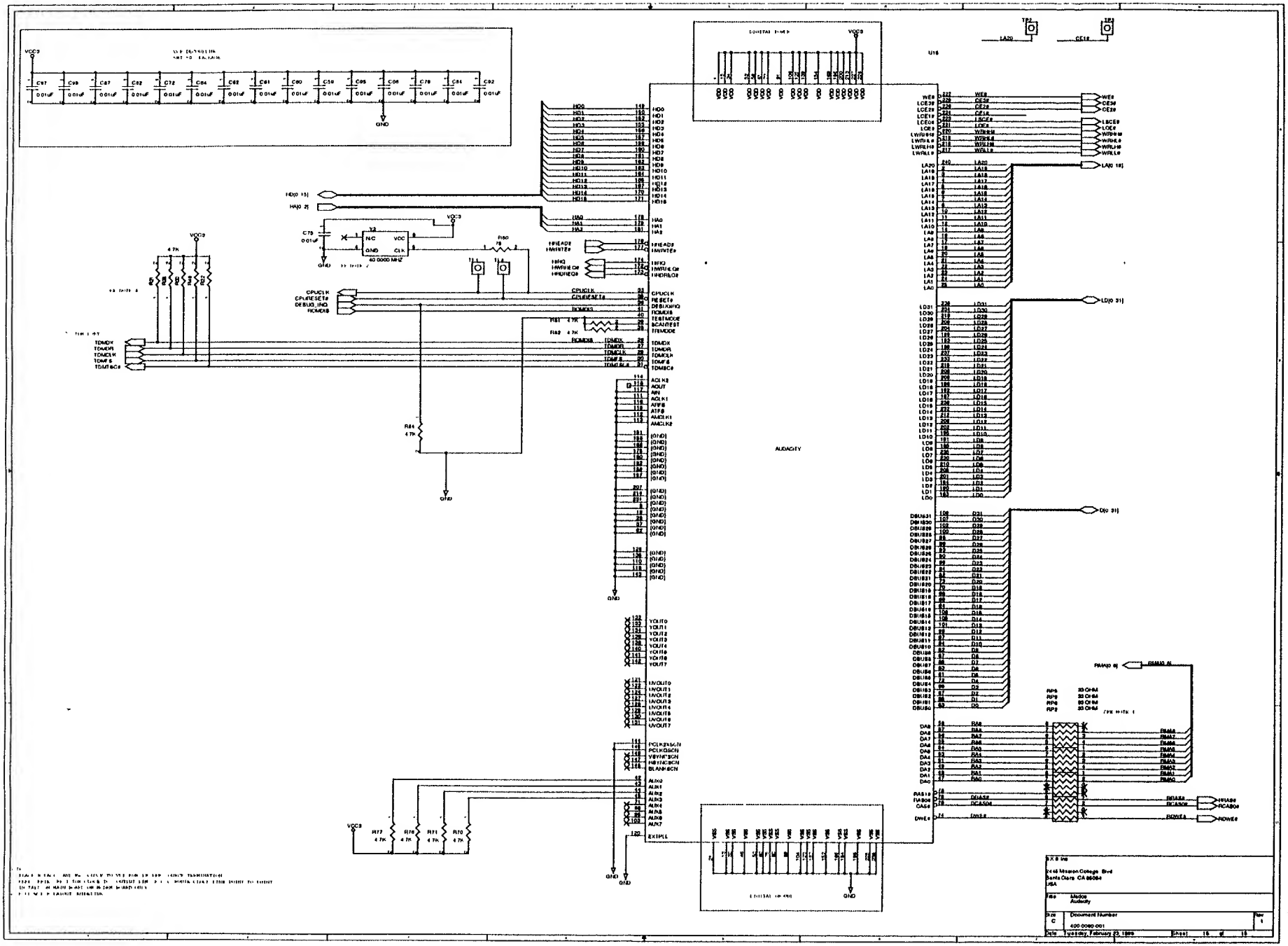


8x8, Inc	
2445 Mission College Blvd Santa Clara, CA 95054 TEL (408) 727-1885 FAX (408) 980-0432	
Title	Madoc SLIC Battery Feed Supply
Size: A	Document Number: 400-0060-001 — Rev: 1
Date: Tuesday, February 23, 1999	Sheet 13 of 15

09740263.121800



0974025e3 .121800


AUDACITY
40.0000 MHz
GND
VCC3
HD[0 15]
HA[0 2]
LA[0 19]
LD[0 31]
D[0 31]
RMA[0 8]
RMD[0 31]
RCASA#
RCASB#
RWE#
RAS0#
RAS1#
EXTPLL
Audacity
Document Number
400-0092-001
Tuesday, February 23, 1999

Appendix 2 - Glue-logic

Confidential - Symphony Labs

```
MODULE Madoc

TITLE 'Madoc 10BaseT to POTS Core Logic'

PLSI PROPERTY 'PULLUP OFF';
PLSI PROPERTY 'EFFORT 3';
PLSI PROPERTY 'STRATEGY AREA';

// ********************************************************************
// SRAM BUS INPUT/OUTPUT
// ********************************************************************
!CE2                            PIN 19;

PLSI    PROPERTY        'SCP CE2 CE2_TO_ETHER_CS';

!OE                             PIN 20;

!WE                             PIN;
PLSI    PROPERTY        'CLK WE CLK2';

LA2, LA3, LA4, LA5, LA6, LA7    PIN 28, 29, 30, 32, 33, 34;
LA19                            PIN 18;

PLSI    PROPERTY        'SCP LA7 LA7_TO_ETHER_CS';
PLSI    PROPERTY        'SCP LA7 LA7_TO_EA1, LA7_TO_EA2, LA7_TO_EA3';
PLSI    PROPERTY        'SCP LA19 LA19_TO_ETHER_CS';
PLSI    PROPERTY        'SCP LA19 LA19_TO_EA1, LA19_TO_EA2, LA19_TO_EA3';

CPUCLK                          PIN;
PLSI PROPERTY 'CLK CPUCLK CLK0';

DEBUG_IRQ                       PIN 23;

D0, D1, D2, D3                  PIN 46, 47, 48, 49      istype 'com';
D4, D5, D6, D7                  PIN 51, 52, 53, 55      istype 'com';

// ********************************************************************
// SRAM BUS INTERMEDIATES
ETHER_BANK              NODE            istype 'collapse';
ETHER_BANK_CE           NODE            istype 'keep';


REGISTER_SPACE          NODE            istype 'collapse';
SYSTEM_ID_ADDRESS       NODE            istype 'collapse';
REGISTER_BANK           NODE            istype 'collapse';
REGISTER_BANK_CE        NODE            istype 'keep';
ADDRESS_0               NODE            istype 'collapse';
ADDRESS_1               NODE            istype 'collapse';
ADDRESS_2               NODE            istype 'collapse';
ADDRESS_3               NODE            istype 'collapse';
ADDRESS_4               NODE            istype 'collapse';
ADDRESS_5               NODE            istype 'collapse';
REGISTER_0              NODE            istype 'collapse';
REGISTER_1              NODE            istype 'collapse';
REGISTER_2              NODE            istype 'collapse';
REGISTER_3              NODE            istype 'collapse';
REGISTER_4              NODE            istype 'collapse';
REGISTER_5              NODE            istype 'collapse';
```

```
// ****************************************************************
// ETHERNET INTERFACES
// ****************************************************************
!ETHER_RD               PIN 40          istype 'invert, reg';
!ETHER_WR               PIN 41          istype 'invert, reg';

!ETHER_CS               PIN 35          istype 'invert,reg';

PLSI    PROPERTY        'ECP ETHER_CS LA7_TO_ETHER_CS, LA19_TO_ETHER_CS, CE2_TO_ETHER_CS';

ETHER_IRQ               PIN 22;
ETHER_IRQ_ENABLE        NODE            istype 'reg, keep';

EA1, EA2, EA3           PIN 24, 26, 27  istype 'reg';

PLSI    PROPERTY        'ECP EA1 LA7_TO_EA1, LA19_TO_EA1';
PLSI    PROPERTY        'ECP EA2 LA7_TO_EA2, LA19_TO_EA2';
PLSI    PROPERTY        'ECP EA3 LA7_TO_EA3, LA19_TO_EA3';

// WAIT STATE COUNTER
WS3, WS2, WS1, WS0      NODE            istype 'reg';

END_COUNT               node            istype 'collapse';

// ****************************************************************
// RESET LOGIC SIGNALS
// ****************************************************************
!POWER_RESET            PIN 42;
!HOST_RESET             PIN 43;
!CPU_RESET              PIN 45                  istype 'com';

ETHER_RESET             PIN 56                  istype 'com';
ETHER_RESET_BIT         NODE                    istype 'reg';

INTERNAL_RESET_INT      NODE                    istype 'keep';
INTERNAL_RESET          NODE                    istype 'collapse';
// ****************************************************************
// SLIC CONTROL AND STATUS BITS
// ****************************************************************

LED_CADENCE_0, LED_CADENCE_1, LED_CADENCE_2, LED_CADENCE_3      NODE                    is
type 'reg';
LED_CADENCE_4, LED_CADENCE_5                                    NODE                    is
type 'reg';
LED_COMMON                                                      PIN 2                   is
type 'reg';
LED_CADENCE_CARRY                                               NODE                    is
type 'com, collapse';

RING_CADENCE_0, RING_CADENCE_1, RING_CADENCE_2                  NODE                    is
type 'reg';
RING_COMMON                                                     NODE                    is
type 'reg';

RNGNG3, RNGNG2, RNGNG1, RNGNG0                                  PIN 74, 76, 72, 73      is
type 'reg';
B2_3, B2_2, B2_1, B2_0                                          PIN 79, 80, 77, 78      is
```

```
type 'com';
B2_BIT3, B2_BIT2, B2_BIT1, B2_BIT0                    NODE                    is
type 'reg';
!STAT3, !STAT2, !STAT1, !STAT0                        PIN 69, 70, 67, 68;

SLIC_LED3, SLIC_LED2, SLIC_LED1, SLIC_LED0            PIN 1, 99, 98, 97       is
type 'com';
SLIC_LED0_0, SLIC_LED0_1                              NODE                    is
type 'reg';
SLIC_LED1_0, SLIC_LED1_1                              NODE                    is
type 'reg';
SLIC_LED2_0, SLIC_LED2_1                              NODE                    is
type 'reg';
SLIC_LED3_0, SLIC_LED3_1                              NODE                    is
type 'reg';

LINK_LED3, LINK_LED2, LINK_LED1, LINK_LED0            PIN 96, 95, 93, 92      is
type 'com';
LINK_LED0_0, LINK_LED0_1                              NODE                    is
type 'reg';
LINK_LED1_0, LINK_LED1_1                              NODE                    is
type 'reg';
LINK_LED2_0, LINK_LED2_1                              NODE                    is
type 'reg';
LINK_LED3_0, LINK_LED3_1                              NODE                    is
type 'reg';


SLICS_ACTIVE                                          PIN 58                  is
type 'reg';
SLIC_0_ACTIVE                                         PIN 17                  is
type 'reg';
// ********************************************************************
// TDM CONTROL AND STATUS BITS
// ********************************************************************
TDM_MAIN_CLK                                          PIN;
PLSI PROPERTY 'CLK TDM_MAIN_CLK CLK1';

!TDMCLK_EN                                           PIN 8                   istype 're
g';
TDMCLK                                                PIN 7                   istype 're
g';
TDMFS_VCP                                             PIN 6                   istype 'co
m';
TDMFS_EXT                                             PIN 57                  istype 'co
m';
TDMDR                                                 PIN 5                   istype 'co
m';
TDMDX                                                 PIN 3;

Q_TDMDR                                               PIN 83;

ALAW_SEL                                              PIN 82                  istype 're
g';
TDM_512                                               NODE                    istype 're
g';
TDM_256                                               NODE                    istype 'co
m, collapse';
```

```
TDM_DIV0, TDM_DIV1, TDM_DIV2                                NODE                    istype 're
g';
TDM_HALF_CARRY                                              NODE                    istype 'co
m, collapse';
TDM_CARRY                                                   NODE                    istype 'co
m, collapse';

BIT_COUNT_0, BIT_COUNT_1, BIT_COUNT_2                       NODE                    istype 're
g';
BIT_COUNT_CARRY                                             NODE                    istype 'co
m, collapse';

SLOT_COUNT_0, SLOT_COUNT_1, SLOT_COUNT_2, SLOT_COUNT_3      NODE                    istype 're
g';
SLOT_COUNT_4, SLOT_COUNT_5                                  NODE                    istype 're
g';
SLOT_COUNT_CARRY                                            NODE                    istype 'co
m, collapse';

FSx3, FSx2, FSx1, FSx0                                      PIN 90, 91, 84, 85      istype 'co
m';

// ********************************************************************
// system ID BITS
// ********************************************************************
SYS_ID_0                NODE            istype 'collapse';
SYS_ID_1                NODE            istype 'collapse';
SYS_ID_2                NODE            istype 'collapse';
SYS_ID_3                NODE            istype 'collapse';
SYS_ID_4                NODE            istype 'collapse';
SYS_ID_5                NODE            istype 'collapse';
SYS_ID_6                NODE            istype 'collapse';
SYS_ID_7                NODE            istype 'collapse';


"Special Constants
// USEFUL ABBREVIATIONS
        x,c,z,u,d,k,f   = .X.,.C.,.Z.,.U.,.D.,.K.,.F.;

// USEFUL SYMBOLS
        TRUE            = -1;
        FALSE           =  0;

// PROGRAMMED FOR THE REQUIRED WAIT STATES
// THIS IS DEPENDENT ON THE TARGET SYSTEM CPUCLK RATE
// (HERE 36MHz ASSUMED) AND ON THE REQUIRED MINIMUM WRITE
// PULSE (HERE 240nS ASSUMED FOR I2C CHIP)
        DELAY_LENGTH    = 2;


// THE NEXT TWO SET THE COUNT LENGTH FOR THE DIVIDERS FOR THE LED AND RING SYSTEMS
        MAX_LED_CADENCE         = 40;
        MAX_RING_CADENCE        = 5;

// BANK ADDRESS DECLARATIONS FOR THE TEST VECTORS (CONVENIENCE)
        REGISTERS_BANK          = 4;
        OTHER_BANK              = 8;
```

Exhibit 1 - 4,218,609

```
// System ID declarations
        ID_VC55_PLUS            = 1;
        ID_VC105_II             = 2;
        ID_VC150_II             = 3;
        ID_VC150_III            = 4;
        ID_VC160                = 4;
        ID_VC105_III            = 5;
        ID_VC110                = 5;
        ID_VST_ISDN             = 6;
        ID_VST_LAN              = 7;
        ID_MADOC                = 10;

"intermediate signal

"Set Declarations
        DELAY_COUNT             = [WS3..WS0];

        EXTENDED_SYSTEM_ID      = [SYS_ID_7..SYS_ID_0];

        D_BUS                   = [D7..D0];

        EAn                     = [EA3..EA1];
        LAn                     = [LA4..LA2];

        REGISTER_0n             = [SLIC_LED0_1, SLIC_LED0_0, B2_BIT0];
        REGISTER_1n             = [SLIC_LED1_1, SLIC_LED1_0, B2_BIT1];
        REGISTER_2n             = [SLIC_LED2_1, SLIC_LED2_0, B2_BIT2];
        REGISTER_3n             = [SLIC_LED3_1, SLIC_LED3_0, B2_BIT3];
        REGISTER_4n             = [TDMCLK_EN,SLIC_0_ACTIVE,SLICS_ACTIVE, TDM_512, ALAW_SEL
, ETHER_RESET_BIT, ETHER_IRQ_ENABLE];
        REGISTER_5n             = [LINK_LED3_1, LINK_LED3_0, LINK_LED2_1, LINK_LED2_0, LIN
K_LED1_1, LINK_LED1_0, LINK_LED0_1, LINK_LED0_0];

        RNGNGn                  = [RNGNG3, RNGNG2, RNGNG1, RNGNG0];
        B2_n                    = [B2_3, B2_2, B2_1, B2_0];
        B2_BITn                 = [B2_BIT3, B2_BIT2, B2_BIT1, B2_BIT0];

        TDM_DIVIDE              = [TDM_DIV2, TDM_DIV1, TDM_DIV0];

        BIT_COUNT               = [BIT_COUNT_2, BIT_COUNT_1, BIT_COUNT_0];

        SLOT_COUNT              = [SLOT_COUNT_5, SLOT_COUNT_4, SLOT_COUNT_3, SLOT_COUNT_2,
 SLOT_COUNT_1, SLOT_COUNT_0];

        LED_CADENCE             = [LED_CADENCE_5, LED_CADENCE_4, LED_CADENCE_3, LED_CADENC
E_2, LED_CADENCE_1, LED_CADENCE_0];
        RING_CADENCE            = [RING_CADENCE_2, RING_CADENCE_1, RING_CADENCE_0];

        BANK_ADDRESS            = [!CE2, LA19, LA7, LA6];


Equations

// ********************************************************************
// TDM GENERATOR AND CONTROL
// ********************************************************************
```

```
TDM_DIVIDE              := TDM_DIVIDE + 1;
TDM_DIVIDE.clk           = TDM_MAIN_CLK;
TDM_DIVIDE.aclr          = INTERNAL_RESET # ETHER_RESET_BIT;

TDM_256                  = !TDM_512;

TDM_HALF_CARRY           = (TDM_512 & (TDM_DIVIDE[0] == 1)) # (TDM_256 & (TDM_DIVIDE[1..0]
 == 1));

TDM_CARRY                = TDM_HALF_CARRY & !TDMCLK;

WHEN TDM_HALF_CARRY
THEN TDMCLK      := !TDMCLK
ELSE TDMCLK      :=  TDMCLK;

TDMCLK.clk               = TDM_MAIN_CLK;
TDMCLK.aclr              = INTERNAL_RESET # ETHER_RESET_BIT;

WHEN TDM_CARRY
THEN BIT_COUNT := BIT_COUNT + 1
ELSE BIT_COUNT := BIT_COUNT;

BIT_COUNT.clk            = TDM_MAIN_CLK;
BIT_COUNT.aclr           = INTERNAL_RESET # ETHER_RESET_BIT;
BIT_COUNT_CARRY          = (BIT_COUNT == -1);

WHEN !(TDM_CARRY & BIT_COUNT_CARRY)
THEN SLOT_COUNT := SLOT_COUNT
ELSE     WHEN SLOT_COUNT_CARRY
         THEN SLOT_COUNT := 0
         ELSE SLOT_COUNT := SLOT_COUNT+1;

SLOT_COUNT.clk           = TDM_MAIN_CLK;
SLOT_COUNT.aclr          = INTERNAL_RESET # ETHER_RESET_BIT;
SLOT_COUNT_CARRY         = (TDM_512 & (SLOT_COUNT == -1)) # (TDM_256 & SLOT_COUNT[4..0] ==
 -1);

TDMFS_VCP                = BIT_COUNT_CARRY & SLOT_COUNT_CARRY;

FSx0                     = BIT_COUNT_CARRY & SLOT_COUNT_CARRY;
FSx1                     = BIT_COUNT_CARRY & (SLOT_COUNT == 0);
FSx2                     = BIT_COUNT_CARRY & (SLOT_COUNT == 1);
FSx3                     = BIT_COUNT_CARRY & (SLOT_COUNT == 2);

// This signbal is now hacked to allow for VCP/VCPex's extra-delayed-bit TDM crud
TDMFS_EXT                = (BIT_COUNT == 6) & SLOT_COUNT_CARRY;

TDMDR                    = Q_TDMDR;
TDMDR.oe                 = (SLOT_COUNT == 0)
                         # (SLOT_COUNT == 1)
                         # (SLOT_COUNT == 2)
                         # (SLOT_COUNT == 3);


// ********************************************************************
// LED COMMON LOGIC
// ********************************************************************
```

```
WHEN !(TDM_CARRY & BIT_COUNT_CARRY & SLOT_COUNT_CARRY)
THEN {
        LED_CADENCE      := LED_CADENCE;
        LED_COMMON       := LED_COMMON;
      }
ELSE WHEN LED_CADENCE < MAX_LED_CADENCE
      THEN {
                LED_CADENCE      := LED_CADENCE +1;
                LED_COMMON       := LED_COMMON;
           }
      ELSE {
                LED_CADENCE      := 0;
                LED_COMMON       := !LED_COMMON;
           };


LED_CADENCE_CARRY                 = (LED_CADENCE == MAX_LED_CADENCE);

LED_CADENCE.clk                   = TDM_MAIN_CLK;
LED_COMMON.clk                    = TDM_MAIN_CLK;

LED_CADENCE.aclr                  = INTERNAL_RESET # ETHER_RESET_BIT;
LED_COMMON.aclr                   = INTERNAL_RESET # ETHER_RESET_BIT;

// ****************************************************************
// RING GENERATOR LOGIC
// ****************************************************************
WHEN !(TDM_CARRY & BIT_COUNT_CARRY & SLOT_COUNT_CARRY & LED_CADENCE_CARRY)
THEN {
        RING_CADENCE     := RING_CADENCE;
        RING_COMMON      := RING_COMMON;
      }
ELSE WHEN RING_CADENCE < MAX_RING_CADENCE
      THEN {
                RING_CADENCE     := RING_CADENCE +1;
                RING_COMMON      := RING_COMMON;
           }
      ELSE {
                RING_CADENCE     := 0;
                RING_COMMON      := !RING_COMMON;
           };


RING_CADENCE.clk                  = TDM_MAIN_CLK;
RING_COMMON.clk                   = TDM_MAIN_CLK;

RING_CADENCE.aclr                 = INTERNAL_RESET # ETHER_RESET_BIT;
RING_COMMON.aclr                  = INTERNAL_RESET # ETHER_RESET_BIT;


// ****************************************************************
// Ethernet INTERFACE LOGIC
// Device space is Bank 2 (CE2), "Bottom Half" (0x00000 - 0x7FFFF)
// (LA19 = 0)
// The Ethernet decodes to the Low 64 bytes of each 128 Byte mirror,
//  with 8 word-wide registers on LongWord Boundaries, mirrored on
//  32-byte boundaries
```

```
// ********************************************************************

ETHER_BANK                = !LA19 & !LA7 & !LA6 & !LA5;
ETHER_BANK_CE             = CE2 & ETHER_BANK;

// Latch & hold address lines
EAn                       := LAn;

ETHER_CS                  := CE2;
ETHER_CS.aset             =  CE2;

ETHER_RD                  := ETHER_BANK_CE & OE;
ETHER_RD.aclr             =  !CE2 # INTERNAL_RESET;


WHEN    !ETHER_WR
THEN    ETHER_WR          := (ETHER_CS) & ETHER_BANK_CE & !OE & !END_COUNT
ELSE    ETHER_WR          := !END_COUNT;

END_COUNT                 = (DELAY_COUNT >= DELAY_LENGTH);

DELAY_COUNT.clk           = CPUCLK;
DELAY_COUNT.aclr          = INTERNAL_RESET;

WHEN    !ETHER_BANK_CE
THEN            DELAY_COUNT     := 0
ELSE    WHEN (END_COUNT # !ETHER_WR)
        THEN    DELAY_COUNT     := DELAY_COUNT
        ELSE    DELAY_COUNT     := DELAY_COUNT +1;

[ETHER_WR, ETHER_RD].clk                =       CPUCLK;
ETHER_CS.clk                            =       CPUCLK;
EAn.clk                                 =       CPUCLK;

ETHER_WR.aclr                           =       INTERNAL_RESET;
EAn.aclr                                =       INTERNAL_RESET;


// ********************************************************************
// RESET LOGIC
// ********************************************************************

INTERNAL_RESET_INT      = HOST_RESET # POWER_RESET;
INTERNAL_RESET          = INTERNAL_RESET_INT;

ETHER_RESET             = INTERNAL_RESET # ETHER_RESET_BIT;

CPU_RESET               = INTERNAL_RESET;

// ********************************************************************
// INTERRUPT CONTROL LOGIC
// ********************************************************************

DEBUG_IRQ               =       (ETHER_IRQ & ETHER_IRQ_ENABLE);

// ********************************************************************
// INTERNAL REGISTERS LOGIC
// ********************************************************************
```

```
// Register Space is Bank 2 (CE2), "High Half" (0x80000 - 0xFFFFF)
// (LA19 = 1)
// Each register is mapped to the Low byte on LongWord boundaries

REGISTER_SPACE          = LA19 & CE2;

SYSTEM_ID_ADDRESS       = LA19 &  LA7 &  LA6 &  LA4 &  LA3 &  LA2;

REGISTER_BANK           = LA19 & !LA7 & !LA6 & !LA5;
REGISTER_BANK_CE        = CE2 & REGISTER_BANK;

ADDRESS_0               = !LA4 & !LA3 & !LA2;
ADDRESS_1               = !LA4 & !LA3 &  LA2;
ADDRESS_2               = !LA4 &  LA3 & !LA2;
ADDRESS_3               = !LA4 &  LA3 &  LA2;
ADDRESS_4               =  LA4 & !LA3 & !LA2;
ADDRESS_5               =  LA4 & !LA3 &  LA2;

REGISTER_0              = REGISTER_BANK_CE & ADDRESS_0;
REGISTER_1              = REGISTER_BANK_CE & ADDRESS_1;
REGISTER_2              = REGISTER_BANK_CE & ADDRESS_2;
REGISTER_3              = REGISTER_BANK_CE & ADDRESS_3;
REGISTER_4              = REGISTER_BANK_CE & ADDRESS_4;
REGISTER_5              = REGISTER_BANK_CE & ADDRESS_5;

// REGISTER 0
REGISTER_0n             := (REGISTER_0 & [D3, D2, D0]) # (!REGISTER_0 & REGISTER_0n);
RNGNG0                  := (REGISTER_0 & (D1 & !(RNGNG1 # RNGNG2 # RNGNG3)))  #
                                         (!REGISTER_0 & RNGNG0);

REGISTER_0n.clk          = !WE;
REGISTER_0n.aclr         = INTERNAL_RESET;

// REGISTER 1
REGISTER_1n             := (REGISTER_1 & [D3, D2, D0]) # (!REGISTER_1 & REGISTER_1n);
RNGNG1                  := (REGISTER_1 & (D1 & !(RNGNG0 # RNGNG2 # RNGNG3)))  #
                                         (!REGISTER_1 & RNGNG1);

REGISTER_1n.clk          = !WE;
REGISTER_1n.aclr         = INTERNAL_RESET;

// REGISTER 2
REGISTER_2n             := (REGISTER_2 & [D3, D2, D0]) # (!REGISTER_2 & REGISTER_2n);
RNGNG2                  := (REGISTER_2 & (D1 & !(RNGNG0 # RNGNG1 # RNGNG3)))  #
                                         (!REGISTER_2 & RNGNG2);

REGISTER_2n.clk          = !WE;
REGISTER_2n.aclr         = INTERNAL_RESET;

// REGISTER 3
REGISTER_3n             := (REGISTER_3 & [D3, D2, D0]) # (!REGISTER_3 & REGISTER_3n);
RNGNG3                  := (REGISTER_3 & (D1 & !(RNGNG0 # RNGNG1 # RNGNG2)))  #
                                         (!REGISTER_3 & RNGNG3);

REGISTER_3n.clk          = !WE;
REGISTER_3n.aclr         = INTERNAL_RESET;

// REGISTER 4
REGISTER_4n             := (REGISTER_4 & [D6..D0]) # (!REGISTER_4 & REGISTER_4n);
REGISTER_4n.clk          = !WE;
REGISTER_4n.aclr         = INTERNAL_RESET;
```

```
// REGISTER 5
REGISTER_5n             := (REGISTER_5 & [D7..D0]) # (!REGISTER_5 & REGISTER_5n);
REGISTER_5n.clk          = !WE;
REGISTER_5n.aclr         = INTERNAL_RESET;

RNGNGn.clk               = !WE;
RNGNGn.aclr              = INTERNAL_RESET;

// *********************************************************************
// SLIC I/O's
// *********************************************************************
WHEN RNGNG0
THEN B2_0 = RING_COMMON
ELSE B2_0 = B2_BIT0;

WHEN RNGNG1
THEN B2_1 = RING_COMMON
ELSE B2_1 = B2_BIT1;

WHEN RNGNG2
THEN B2_2 = RING_COMMON
ELSE B2_2 = B2_BIT2;

WHEN RNGNG3
THEN B2_3 = RING_COMMON
ELSE B2_3 = B2_BIT3;

SLIC_LED0       = (!SLIC_LED0_1 & !SLIC_LED0_0 & LED_COMMON)
                # (!SLIC_LED0_1 &  SLIC_LED0_0 & TRUE)
                # ( SLIC_LED0_1 & !SLIC_LED0_0 & !LED_COMMON)
                # ( SLIC_LED0_1 &  SLIC_LED0_0 & FALSE);

SLIC_LED1       = (!SLIC_LED1_1 & !SLIC_LED1_0 & LED_COMMON)
                # (!SLIC_LED1_1 &  SLIC_LED1_0 & TRUE)
                # ( SLIC_LED1_1 & !SLIC_LED1_0 & !LED_COMMON)
                # ( SLIC_LED1_1 &  SLIC_LED1_0 & FALSE);

SLIC_LED2       = (!SLIC_LED2_1 & !SLIC_LED2_0 & LED_COMMON)
                # (!SLIC_LED2_1 &  SLIC_LED2_0 & TRUE)
                # ( SLIC_LED2_1 & !SLIC_LED2_0 & !LED_COMMON)
                # ( SLIC_LED2_1 &  SLIC_LED2_0 & FALSE);

SLIC_LED3       = (!SLIC_LED3_1 & !SLIC_LED3_0 & LED_COMMON)
                # (!SLIC_LED3_1 &  SLIC_LED3_0 & TRUE)
                # ( SLIC_LED3_1 & !SLIC_LED3_0 & !LED_COMMON)
                # ( SLIC_LED3_1 &  SLIC_LED3_0 & FALSE);

// *********************************************************************
// SYSTEM STATUS I/O's
// *********************************************************************
LINK_LED0       = (!LINK_LED0_1 & !LINK_LED0_0 & LED_COMMON)
                # (!LINK_LED0_1 &  LINK_LED0_0 & TRUE)
                # ( LINK_LED0_1 & !LINK_LED0_0 & !LED_COMMON)
                # ( LINK_LED0_1 &  LINK_LED0_0 & FALSE);

LINK_LED1       = (!LINK_LED1_1 & !LINK_LED1_0 & LED_COMMON)
                # (!LINK_LED1_1 &  LINK_LED1_0 & TRUE)
```

```
                # ( LINK_LED1_1 & !LINK_LED1_0 & !LED_COMMON)
                # ( LINK_LED1_1 &  LINK_LED1_0 & FALSE);

LINK_LED2       = (!LINK_LED2_1 & !LINK_LED2_0 & LED_COMMON)
                # (!LINK_LED2_1 &  LINK_LED2_0 & TRUE)
                # ( LINK_LED2_1 & !LINK_LED2_0 & !LED_COMMON)
                # ( LINK_LED2_1 &  LINK_LED2_0 & FALSE);

LINK_LED3       = (!LINK_LED3_1 & !LINK_LED3_0 & LED_COMMON)
                # (!LINK_LED3_1 &  LINK_LED3_0 & TRUE)
                # ( LINK_LED3_1 & !LINK_LED3_0 & !LED_COMMON)
                # ( LINK_LED3_1 &  LINK_LED3_0 & FALSE);


// *********************************************************************
// BOARD ID VALUES
// *********************************************************************
EXTENDED_SYSTEM_ID      =       ID_MADOC;


// *********************************************************************
// DATA BUS OUTPUT
// *********************************************************************

D_BUS.OE        = (REGISTER_SPACE) & OE;


D_BUS           = (REGISTER_SPACE &
                        (REGISTER_BANK &
                                (
                                  (ADDRESS_0    & [0, 0, 0, STAT0, SLIC_LED0_1, SLIC_LED0_0
, RNGNG0, B2_BIT0])
                                # (ADDRESS_1    & [0, 0, 0, STAT1, SLIC_LED1_1, SLIC_LED1_0
, RNGNG1, B2_BIT1])
                                # (ADDRESS_2    & [0, 0, 0, STAT2, SLIC_LED2_1, SLIC_LED2_0
, RNGNG2, B2_BIT2])
                                # (ADDRESS_3    & [0, 0, 0, STAT3, SLIC_LED3_1, SLIC_LED3_0
, RNGNG3, B2_BIT3])
                                # (ADDRESS_4    & [0, TDMCLK_EN, SLIC_0_ACTIVE, SLICS_ACTIV
E, TDM_512, ALAW_SEL, ETHER_RESET_BIT, ETHER_IRQ_ENABLE])
                                # (ADDRESS_5    & [LINK_LED3_1, LINK_LED3_0, LINK_LED2_1, L
INK_LED2_0, LINK_LED1_1, LINK_LED1_0, LINK_LED0_1, LINK_LED0_0])
                                )
                        )
                    # (SYSTEM_ID_ADDRESS & EXTENDED_SYSTEM_ID)
                );
END
```

Appendix 3 - Contacts

Sales

North America

Chris Peters – Vice President Sales
Phone +1 408 654 0811
Email chrisp@8x8.com

Huw Rees – Director North America Sales
Phone +1 408 654 0842
Email huwr@8x8.com

Asia

Doug Bailey - Director Asian OEM Sales
Phone +1 408 654 0860
Email dougb@8x8.com

Virginia Lee – Director Asian Sales
Phone +1 408 654 0870
Email virginial@8x8.com

Joseph Chen – Manager Asian Sales
Phone +1 408 654 0870
Email josephc@8x8.com

Fax +1 408 980 0432

Europe

David Mills – Manager European OEM sales
Phone +44 1628 402800
Fax +44 1628 402829
Email davidm@8x8.com

Support

Robert Simmons – Manager Applications Engineering
Phone +1 408 654 0993
Fax 408 980 0432
Email robert@8x8.com

General Email addresses
apps@8x8.com
oem@8x8.com

Application Engineering web-site
www.8x8.com/software

Confidential - 8x8 Inc.

Appendix 4 - Glossary

SLIC	SUSCRIBER LINE INTERFACE CIRCUIT
PSTN	PUBLIC SWITCHED TELEPHONE NETWORK
POTS	PLAIN OLD TELEPHONE SYSTEM (PSTN)
ISDN	INTEGRATED SYSTEMS DIGITAL NETWORK
LAN	LOCAL AREA NETWORK
DSL	DIGITAL SUSCRIBER LINE
MGCP	MEDIA GATEWAY CONTROL PROTOCOL
SGCP	SIMPLE GATEWAY CONTROL PROTOCOL
SOHO	SMALL OFFICE / HOME OFFICE
H.323	ITU STANDARD FOR PACKET SWITCHED MEDIA
ITU	INTERNATIONAL TELECOMMUNICATIONS UNION
PBN	PACKET BASED NETWORK
CHI	CONCENTRATION HIGHWAY INTERFACE
CPE	CUSTOMER PREMISE EQUIPMENT
EPLD	ELECRONICALLY PROGRAMMABLE LOGIC DEVICE
ITP	INTERNET TELEPHONY PROCESSOR
MAC	MEDIA ACCESS CONTROLLER
MVIP	MULTI-VENDOR INDEPENDENT PROTOCOL
PCM	PULSE CODE MODULATION
REN	RINGER EQUIVALENCE NUMBER
TDM	TIME DIVISION MULTIPLEXED
VoIP	VOICE OVER INTERNET PROTOCOL
CO	CENTRAL OFFICE
DTMF	DUAL TONE MULTIPLE FREQUENCY
FSK	FREQUENCY SHIFT KEYING
IP	INTERNET PROTOCOL
RTP	REAL TIME PROTOCOL
RTCP	REAL TIME CONTROL PROTOCOL

Confidential - Intel